CI4A: Semantic Component Interfaces for Agents Empowering Web Automation

Zhi Qiu[1] **Jiazheng Sun**[2] **Chenxiao Xia**[1]
Jun Zheng[1,3*] **Xin Peng**[2]
[1] Beijing Institute of Technology, Beijing, China; [2] Fudan University, Shanghai, China
[3] Key Laboratory of Intelligent Networking Technology for Social Governance, Ministry of Industry and Information Technology
qiuzhi@bit.edu.cn; jzsun24@m.fudan.edu.cn; chenxiao_xia@bit.edu.cn; zhengjun@bit.edu.cn; pengxin@fudan.edu.cn

Abstract

While Large Language Models demonstrate remarkable proficiency in high-level semantic planning, they remain limited in handling fine-grained, low-level web component manipulations. To address this limitation, extensive research has focused on enhancing model grounding capabilities through techniques such as Reinforcement Learning. However, rather than compelling agents to adapt to human-centric interfaces, we propose constructing interaction interfaces specifically optimized for agents. This paper introduces Component Interface for Agent (CI4A), a semantic encapsulation protocol that abstracts the complex interaction logic of UI components into a set of unified tool primitives accessible to agents. We implemented CI4A within Ant Design, an industrial-grade front-end framework, covering 23 categories of commonly used UI components. Furthermore, we developed a hybrid agent featuring an action space that dynamically updates according to the page state, enabling flexible invocation of available CI4A tools. Leveraging the CI4A-integrated Ant Design, we refactored and upgraded the WebArena benchmark for infrastructure-enabled evaluation. Experimental results demonstrate that the CI4A-based agent outperforms the evaluated approaches, achieving a task success rate of 86.3% in the CI4A-integrated environment, alongside substantial improvements in execution efficiency.

1 Introduction

With the evolution of Large Language Models, Web Agents have demonstrated immense potential in handling complex, open-ended tasks [34, 21, 37]. While they excel in information-intensive scenarios with heavy reasoning but light interaction like Deep Research [27, 36], they struggle with operation-intensive tasks requiring precise UI manipulation and multi-step state changes. Constrained by limited precision and robustness at the interaction level, their task success rates remain significantly below the standards required for industrial applications.

The fundamental cause lies in the fact that modern web pages are constructed for human users capable of processing continuous visual feedback and executing millisecond-level motor control, whereas agents are constrained by a discrete, high-latency Observation-Reasoning-Action loop. On the information side, web interfaces universally adopt human-centric Progressive Disclosure mechanisms (e.g., lazy loading, collapsible panels). This forces agents to expend substantial reasoning resources on tedious exploratory actions merely to complete the decision-making context [8, 14, 39]. On the execution side, interaction modalities centered on visual coordinates and low-level events confine

*Corresponding author.

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

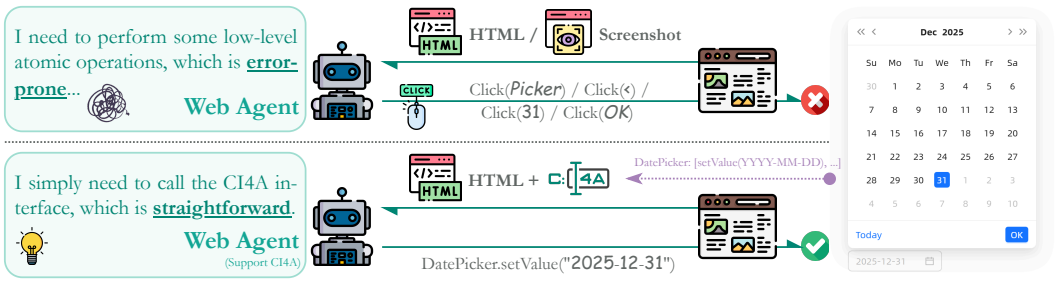

Figure 1: **Fragile Atomic Operations vs. Robust CI4A.** In contrast to traditional methods entangled in error-prone multi-step chains for date selection, CI4A collapses complex logic into **deterministic single-step invocations**, eliminating the uncertainty of long-sequence execution.

agents to a low-level action space, compelling them to decompose high-dimensional user intents into lengthy chains of atomic operations [7, 12]. Given the agent's discontinuous operational mode, such long sequences not only significantly reduce execution efficiency but also make the process highly susceptible to error accumulation from minor low-level missteps, leading to the collapse of the overall high-level task.

To address this challenge, existing research primarily focuses on two directions: first, enhancing internal capabilities by leveraging supervised fine-tuning [17, 9, 15] or Reinforcement Learning [18, 38, 3] to improve the agent's grounding precision; second, constructing High-level Skills, which attempts to encapsulate complex low-level interaction sequences into callable tools (e.g., search, sort) to simplify the operation space [35, 22, 11]. However, the former fails to eliminate the dependency on the fragile DOM, while the latter often requires complex reverse engineering [23] or heuristic rules to synthesize skills [30], making generality difficult to ensure.

Rather than patching or re-abstracting human-centric interfaces [1], we propose reconstructing the interaction paradigm from the source to achieve native alignment between interface semantics and agent reasoning logic. Modern web applications are universally based on modular UI component architectures, which naturally encapsulate high-level functional semantics. This offers an ideal entry point to transcend low-level DOM or GUI limitations and reshape a component-centric interaction paradigm (Figure 1).

To this end, we introduce Component Interface for Agent (CI4A), a semantic encapsulation protocol designed to bridge the gap between high-level task intents and low-level web interactions. CI4A abstracts complex UI components into a set of unified, standardized **Tool Primitives** optimized for agent invocation patterns. By exposing structured interfaces for state retrieval and functional operations, CI4A allows agents to bypass the noisy and fragile low-level DOM, enabling efficient interaction directly at the component level with explicit semantics.

Building upon CI4A, we developed Eous, a hybrid-architecture agent capable of dynamically extending its action space based on the current page state to include and utilize tools exposed by CI4A components. We instantiated CI4A within the industrial-grade Ant Design UI framework, covering 23 categories of commonly used UI components, and utilized this implementation to refactor the Playground scenarios of the WebArena dataset for evaluation. Experimental results indicate that the CI4A-based Eous achieved a task success rate of 86.3% across 182 tasks. It reduces the average number of interaction steps from 10.4 to 4.7 and surpasses AgentOccam (70.3%) in the CI4A-integrated environment. These results demonstrate the substantial contribution of CI4A to improving agent performance. The contributions of this paper are summarized as follows.

- We propose **CI4A**, a semantic encapsulation protocol that transforms UI components into standardized tool primitives, natively aligning interface semantics with agent reasoning logic. We provide **AntDX**, a reference implementation covering 23 core Ant Design components to validate its generality.
- We design **Eous**, an agent that dynamically expands its component-level action space according to the page context and combines high-level semantic tools with low-level operation fallbacks. This layered strategy enables atomic execution of high-dimensional intents, significantly enhancing both effectiveness and efficiency.

- We reconstruct the WebArena Playground environment using the reference implementation while preserving task semantics. Experiments show that **CI4A improves the task success rate and execution efficiency of Eous**.

Our code is available at `https://github.com/ElysiaCyrene/CI4A-public`.

2 Related Work

Perception and Grounding in Web Agents. Existing work primarily explores two paradigms for environment perception and instruction grounding. Structure-based methods, such as WebArenaBase [41] and Mind2Web [6], use Accessibility Trees or HTML text to perceive webpages and ground actions by matching element identifiers or attributes. AgentOccam [33] improves Web agent performance by adjusting observation and action spaces. Vision-based methods introduce screenshots to capture rendering details, with two main grounding mechanisms. The first is direct coordinate regression: SeeClick [4], for example, uses an end-to-end model to predict click locations directly. The second is visual prompting: WebVoyager [10] combines screenshots with Set-of-Mark (SoM) annotations, overlaying numerical labels to turn localization into a discrete identifier-selection problem. VisualWebArena [13] provides a benchmark for evaluating agents on visually grounded web tasks. Specialized models such as OmniParser [20] have also emerged to help visual models parse screen elements more accurately. However, both paradigms are constrained by the progressive disclosure mechanisms of the modern Web: agents cannot directly perceive implicit states in lazy-loaded content or collapsed interfaces and must therefore resort to inefficient physical exploration. End-to-end visual methods also continue to face robustness challenges, including hallucinations and coordinate drift [24].

Action Spaces and Skill Acquisition. The granularity of action abstraction determines interaction efficiency. Mainstream agents still commonly rely on atomic actions that simulate human operations, such as `click`, `type`, and `scroll` [33]. Although broadly applicable, this low-level interaction paradigm produces lengthy decision chains that are highly susceptible to error accumulation. To improve efficiency, SkillWeaver [40] generates reusable APIs through website exploration and practice, ASI [31] induces and verifies programmatic skills from successful interaction experiences, and WALT [23] constructs callable tools by reverse-engineering website functionality. However, these methods fundamentally encapsulate functionality at the business level, providing post-hoc abstractions over existing interfaces. Such abstractions incur high construction costs, have limited generalizability, and remain fragile: even minor changes to frontend layouts or APIs can invalidate these inferred skills. Complementary work focuses on agent coordination: CAMEL [16] explores role-based cooperation, while Mobile-Agent-v2 [29] coordinates planning, decision-making, and reflection. These architectural approaches address a different aspect of agent performance from CI4A's component-level interaction protocol.

Paradigms of Agent-Web Interaction. Traditional Web user interfaces are designed for human visual cognition, whereas Web APIs target developers; neither is sufficiently adapted to agents' discrete decision-making characteristics. SWE-agent [32] demonstrates the importance of agent-computer interface design in software engineering. To address the interface mismatch on the Web, Lù et al. [19] propose the concept of an Agentic Web Interface (AWI), calling for interaction standards specifically optimized for agents. This vision echoes WAI-ARIA's approach to enhancing semantic accessibility [5]. VOIX [25] uses declarative `<tool>` and `<context>` elements through which application developers define the states and business capabilities accessible to agents for a particular website and connect them to application logic. Although its developer study demonstrates the feasibility of prototype development, this adaptation must be repeated across websites and business functions. A gap therefore remains for a concrete protocol implementation that integrates deeply with the modern component-based frontend ecosystem while requiring minimal intrusion into existing applications. CI4A addresses this gap through a reusable component-level interaction protocol.

3 Methodology

3.1 The CI4A Protocol

The objective of CI4A is to establish a standard Web interaction protocol tailored for agents. This protocol aims to bridge the semantic gap between human-computer interfaces and agent cognition. By

defining the semantic interfaces that UI components expose to agents and standardizing the interaction patterns for environment perception and action execution, CI4A transforms Web interaction from fragile reverse engineering based on rendered results into a robust paradigm of forward invocation based on component instances.

3.1.1 UI Component Interface

Within the CI4A protocol, we abstract the interaction capabilities exposed by any UI component C into a semantic triplet $\langle \mathcal{S}, \mathcal{T}, \mathcal{M} \rangle$. This definition constitutes a static contract governing the communication between the agent and the component:

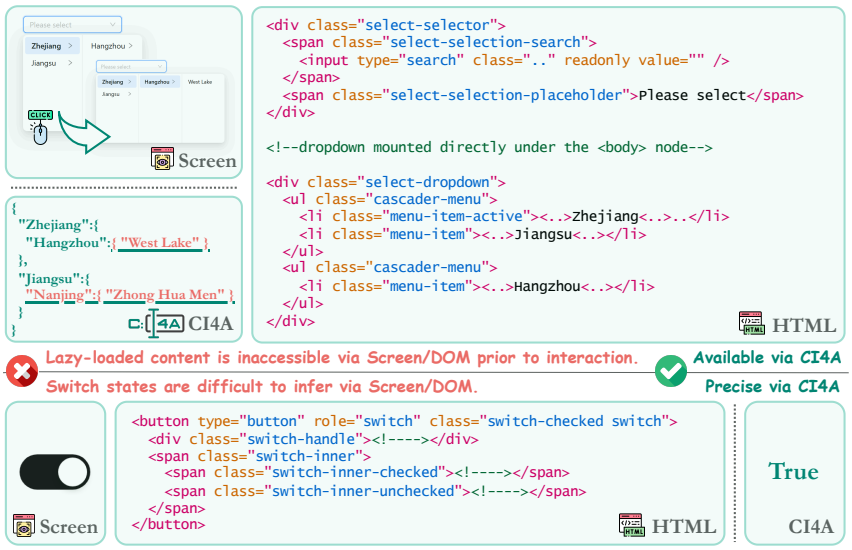

Figure 2: CI4A's Semantic State View ensures accurate state perception for hidden lazy-loaded data and ambiguous DOM states.

Semantic State View $\mathcal{S}$ acts as a direct mapping of the component's internal data model, enabling the agent to bypass the rendering layer and access pure business data directly from memory (e.g., Vue's reactive objects or React's State/Props). This mechanism reduces the Context Encoding Cost from $O(N)$, which scales with DOM tree depth, to $O(1)$, which is decoupled from rendering details. Consequently, it enhances the completeness and fidelity of perception (Figure 2). Specifically, it enables the agent to penetrate Progressive Disclosure mechanisms, such as lazy loading, to acquire a complete view of the data while accessing raw business values. This effectively eliminates the noise and errors typically introduced by DOM parsing and CV recognition.

Executable Toolset $\mathcal{T}$ defines a collection of Atomic Operation Primitives that encapsulate the logic for component state mutation. Whether derived from the component's native public methods or the direct manipulation of internal states, $\mathcal{T}$ uniformly abstracts these operations as standard function calls at the protocol level. By collapsing error-prone, low-level multi-step operation chains into single atomic invocations, this mechanism mitigates hallucinations and cascading errors arising from long-sequence execution. Furthermore, $\mathcal{T}$ establishes an optimal granularity between low-level DOM events and high-level business logic. This abstraction shields the agent from tedious execution details while ensuring the tools remain generalizable and reusable across scenarios.

Interaction Metadata $\mathcal{M}$ defines a Structured Contract for tool invocation, explicitly describing parameter data types, allowed enumeration values, and required-field constraints based on component attributes. This mechanism addresses the issue of constraint invisibility, where restrictions are imperceptible solely through visual or DOM cues. Such invisibility often leads to execution failures because Large Models lack prior knowledge of input formats. By dynamically transforming $\mathcal{M}$ into Schema definitions within the Prompt, CI4A effectively bridges the agent's knowledge gap of business rules. This enables the agent to generate valid actions based on explicit knowledge of tool parameters and their constraints, thereby preventing blind trial-and-error driven by information insufficiency.

3.1.2 Agent-UI Interaction Patterns

In the CI4A protocol, we standardize the entire process by which the agent utilizes the aforementioned interfaces to perceive the environment and execute actions into a set of **canonical dynamic interaction patterns**. This definition establishes the dynamic mechanism governing the collaboration between the agent and the runtime Web UI environment (Figure 3):

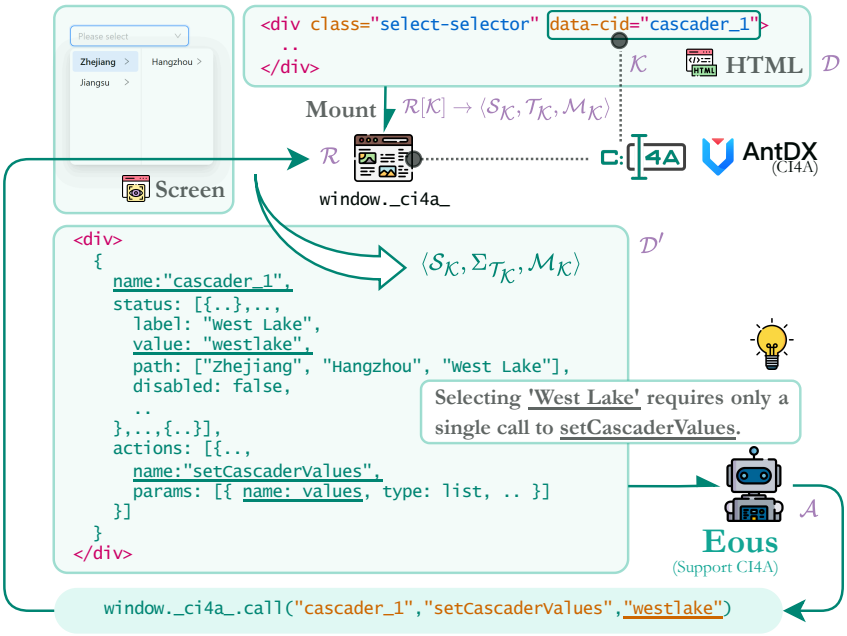

Figure 3: **CI4A workflow.** Components register with a global registry, allowing agents to retrieve the triplet $\langle \mathcal{S}_\mathcal{K}, \Sigma_{\mathcal{T}_\mathcal{K}}, \mathcal{M}_\mathcal{K} \rangle$ for hybrid view construction. The agent then determines tool invocations and their parameters from this view and executes the tools through the registry.

Hybrid View Construction. To balance the efficient perception of spatial Layout Structure and component semantics, CI4A proposes a hybrid view construction strategy that transforms the raw DOM tree $\mathcal{D}$ into a semantic hybrid tree $\mathcal{D}'$. During the component rendering stage, UI component libraries implementing CI4A must assign a unique identifier $\mathcal{K}$ to each component instance and explicitly annotate it on the corresponding DOM root node. This process remains transparent to existing Web development workflows and does not interfere with standard DOM debugging or parsing. For an agent $\mathcal{A}$ supporting CI4A, the strategy mandates specific behaviors when traversing $\mathcal{D}$. For unmarked structural nodes (such as div or span tags responsible for layout), $\mathcal{A}$ retains their hierarchical relationships to preserve spatial context. Conversely, for nodes annotated with $\mathcal{K}$, $\mathcal{A}$ replaces the underlying DOM with the component's corresponding semantic description node, which comprises the semantic state view $\mathcal{S}$, the function signatures $\Sigma_\mathcal{T}$ of the executable tool set, and the interaction metadata $\mathcal{M}$.

Global Registration & Indexing. To enable the agent to invoke tools $\mathcal{T}$ declared within the view, CI4A mandates that the runtime environment maintain a global registry $\mathcal{R}$. This registry $\mathcal{R}$ must be exposed within the browser's JavaScript runtime global object (e.g., window), serving as a conduit for communication between the agent and component instances. Upon component mounting, the instance $\mathcal{K}$ automatically establishes the mapping $\mathcal{R}[\mathcal{K}] \rightarrow \langle \mathcal{S}_\mathcal{K}, \mathcal{T}_\mathcal{K}, \mathcal{M}_\mathcal{K} \rangle$, which is subsequently removed upon unmounting. The agent $\mathcal{A}$ can access $\mathcal{R}$ via the global object to directly index component instances using the key $\mathcal{K}$, thereby retrieving $\langle \mathcal{S}_\mathcal{K}, \Sigma_{\mathcal{T}_\mathcal{K}}, \mathcal{M}_\mathcal{K} \rangle$ or accessing the execution handle $t \in \mathcal{T}_\mathcal{K}$.

Deterministic Execution. To achieve determinism and observability in the interaction process, CI4A requires the agent $\mathcal{A}$ to synthesize action commands t based primarily on the action definitions provided by $\Sigma_\mathcal{T}$ and the constraints defined in $\mathcal{M}$, subsequently issuing an invocation request to $\mathcal{R}$. Simultaneously, CI4A mandates that $\mathcal{R}$ perform runtime parameter validation prior to instruction execution. If a parameter violation is detected, the component is disabled, or $\mathcal{K}$ is invalid, $\mathcal{R}$ must

terminate execution and return a structured exception. This mechanism facilitates the construction of a rapid feedback loop devoid of visual model intervention. It effectively resolves the confirmation dilemma faced by traditional visual agents, where determining whether an action was effective remains difficult even if the correct element was clicked.

3.2 AntDX: A Reference Implementation of CI4A Based on the Ant Design UI Library

Ant Design [28] stands as one of the de facto standards for enterprise Web development. To validate the completeness of the CI4A protocol, we developed a reference implementation named **AntDX**. By applying **Intrusive Instrumentation** to the source code of 23 core components, AntDX assumes control over the component lifecycle and event streams, achieving a deep integration of the protocol logic.

3.2.1 Core Instrumentation Components

Global Registrar. Exposed via *window.__ci4a__*, this component serves as the unified gateway for interactions between the Agent and UI components. It dynamically maintains the Transceivers for all active components and routes requests to provide two core capabilities: **a) State Provisioning ($get\mathcal{S}tatus()$):** Supplies the business semantic state $\mathcal{S}$, tool signatures $\Sigma_{\mathcal{T}}$, and metadata $\mathcal{M}$. **b) Action Execution ($call\mathcal{T}ool()$):** Triggers deterministic business logic within the component.

Component Transceiver. An execution unit embedded within the component, the Transceiver establishes a semantic bridge between the low-level instance and the Agent. It comprises three distinct modules:

- **Auto-Register:** Automatically manages the mapping between the instance and the Global Registrar during the Mount/Unmount lifecycle stages. Additionally, it binds a unique identifier to the data-cid attribute of the DOM.
- **Props Listener:** Reactively monitors component Props. Upon receiving a request for the business semantic state, it filters out noisy fields based on a whitelist, subsequently generating and returning a pure semantic view that encapsulates the core business state.
- **Dispatcher:** Monitors instructions from the Global Registrar and orchestrates their dispatch. Semantic state requests are routed to the Props Listener. Action execution requests are prioritized for forwarding to standard event handlers (e.g., `update:value`), allowing the component to be driven without disrupting the normal event flow.

3.2.2 Implementation of UI Component Interfaces

Based on the distinct roles components play in Agent decision-making and execution during Web tasks, we categorize them into three types: Navigation (e.g., Menu), Data Entry (e.g., DatePicker), and Data Display (e.g., Table). For each category, we implemented a mapping from the raw UI to the semantic triplet $\langle \mathcal{S}, \mathcal{T}, \mathcal{M} \rangle$.

Semantic State View $\mathcal{S}$. Navigation components expose reachable routes or step nodes. Data Entry components provide field identifiers, input guidance, and the set of currently selected or input values. Data Display components present structured data content within the viewport (e.g., table headers and row data).

Executable Tool Set $\mathcal{T}$. Navigation components provide `navigateTo` to implement direct transitions. Data Entry components offer `setValue` and `submit` to efficiently complete data entry and trigger validation. Data Display components provide `sort` and `filter` to adjust data presentation and precisely screen for required information.

Interaction Metadata $\mathcal{M}$. This comprises metadata objects that are manually constructed and injected at compile time (exhaustively defining field types, etc.). For Data Entry components, $\mathcal{M}$ additionally includes input constraints calculated at runtime, such as date formats and numerical ranges.

3.3 Eous: A Hybrid Action Space Agent

To validate the effectiveness of the CI4A protocol, we developed Eous based on Fairy [26], a GUI agent framework featuring a typical Planner-Decider-Reflector closed loop. Benefiting from Fairy's

highly decoupled software design pattern, we seamlessly migrated its originally mobile-oriented perception module to the Web environment. Furthermore, we refactored the Action Decider to introduce a Hybrid Action Space mechanism. This mechanism enables Eous to transcend the limitations of low-level atomic operations by dynamically invoking the component-level semantic tools defined in Section 3.2. Specifically, Eous implements a semantic-atomic hybrid architecture within two critical phases: perception and decision-making.

In the perception phase, Eous introduces a modality-agnostic semantic injection mechanism that is compatible with and enhances mainstream structural and visual perception paradigms:

For Structural Input, The agent scans the DOM / Accessibility tree for instrumented nodes $\mathcal{K}$ bearing the data-cid attribute. It then retrieves the semantic triplet $\langle \mathcal{S}, \Sigma_{\mathcal{T}}, \mathcal{M} \rangle$ via *window.__ci4a__* and directly replaces the original subtree. This semantic-structural hybrid view eliminates redundant node noise while preserving the original page topology.

For Visual Input, Eous integrates Set-of-Mark (SoM) technology to overlay visual ID masks onto interactive elements. It injects the retrieved semantic triplet $\langle \mathcal{S}, \Sigma_{\mathcal{T}}, \mathcal{M} \rangle$ into the Prompt as supplementary context strongly correlated with the visual ID. In this configuration, $\mathcal{S}$ functions as a dynamic legend that elucidates implicit states. Meanwhile, $\Sigma_{\mathcal{T}}$ and $\mathcal{M}$ bind executable semantic operation contracts, allowing the agent to bypass low-level coordinate operations and directly plan high-level component actions.

In the decision-making phase, Eous's action space is defined as the union $a = a_{\Sigma_{\mathcal{T}}} \cup a_{atomic}$. We define a unified primitive, $call(k, \tau, p)$, where $k \in \mathcal{K}$ represents the target component, $\tau \in \Sigma_{\mathcal{T}_k}$ denotes the tool signature, and p is the set of parameters that must satisfy the constraints of metadata $\mathcal{M}_{\mathcal{K}}$. Execution adheres to a Semantic First strategy. This strategy prioritizes the invocation of standardized tools $a_{\Sigma_{\mathcal{T}}}$ exposed by components (e.g., `navigateTo`, `setValue`). The agent adaptively falls back to low-level WebDriver atomic operations (e.g., `click`, `type`) only when applicable semantic tools are absent or execution is obstructed. This approach strikes a balance between interaction efficiency and generality.

4 Experiment

4.1 Setup

▷ **Benchmark.** We selected WebArena as our primary benchmark to evaluate the CI4A protocol within a highly realistic and reproducible environment. However, validating a new interaction protocol presents a cold start dilemma far more severe than validating a new algorithm. Existing benchmarks, such as WebArena, are tailored for agents relying on standard DOM or visual interactions, and their underlying UI component libraries cannot be swapped directly. Consequently, we performed a semantic-preserving refactoring of the WebArena frontend using the standard Ant Design library. The refactored pages strictly maintain the original layout, content, and business logic, replacing native components solely with their Ant Design equivalents, which ensures that visual and logical flows remain unchanged. This design minimizes interference arising from environmental discrepancies, thereby ensuring a fair experimental comparison.

Building on this foundation, and leveraging the API consistency between AntDX and the standard library, we seamlessly integrated AntDX components via dependency replacement without modifying any business code. This implementation validates the low intrusiveness of the CI4A protocol and demonstrates that it imposes zero additional development costs on existing frontend engineering systems. More details are provided in Appendix A.1.

▷ **Baseline.** We selected two groups of representative baselines to evaluate task performance alongside Eous in the CI4A-integrated environment:

- **Unimodal Structural Perception:** WebArenaBase and AgentOccam (text-only mode). Both are powered by GPT-5 and establish the performance benchmark for agents relying on the Accessibility Tree.
- **Multimodal Perception:** AgentOccam (standard mode, GPT-5) and the end-to-end framework MidScene (powered by native Seed-1.8 [2]). These establish benchmarks for visually enhanced and end-to-end agents.

Table 1: Main results in the CI4A-integrated WebArena environment (S: structural input; V: visual input). Eous(V) achieves 86.3% success with 4.7 average steps.

Framework	Model	Accessibility Tree	Screenshot	SR (%)	Avg. Steps
WebArenaBase [41]	GPT-5	✓	✗	26.4	10.0
AgentOccam (S) [33]	GPT-5	✓	✗	69.2	9.4
AgentOccam (V) [33]	GPT-5	✓	✓	70.3	10.4
MidScene [42]	Seed-1.8	✗	✓	61.5	13.0
Eous (S)	GPT-5	✓	✗	**75.3**	**4.2**
Eous (V)	GPT-5	✗	✓	**86.3**	**4.7**

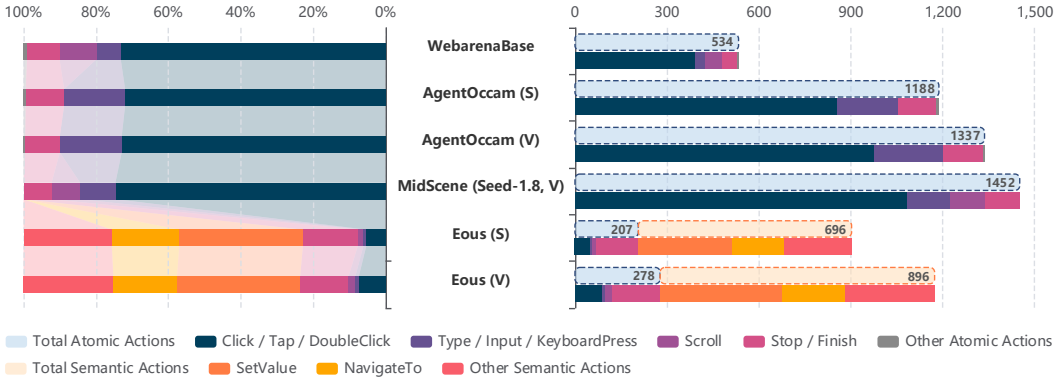

Figure 4: **Action space distribution of Eous vs. Baselines.** Compared to baselines that rely on lengthy low-level atomic actions, Eous leverages high-level semantic tools via CI4A to significantly collapse physical interaction chains, thereby reducing total steps.

Correspondingly, we configured Eous (GPT-5) in two modes: structural input mode (Accessibility Tree infused with semantic triplets) and visual input mode (SoM combined with semantic triplet context supplementation).

All GPT-5 experiments were conducted via the OpenAI API (Temperature=0, Context=128k). For all agents, the maximum number of interaction steps per task was set to 30; exceeding this limit constitutes a failure. Detailed discussions of the benchmark and experimental settings are provided in Appendix A.3 and Appendix A.4.

▷ **Metrics.** We adopt a dual-metric evaluation system:

- **Success Rate(SR):** Measures effectiveness. Adhering to WebArena's Functional Correctness standard, this validates whether the final page or database state aligns with the Oracle's expectation upon task completion.
- **Average Steps(Avg. Steps):** Measures efficiency. By quantifying the decision turns, this metric evaluates the chain collapse effect of high-level semantic tools relative to low-level atomic operations.

4.2 Main Results

We report the overall performance of Eous in the CI4A-integrated WebArena environment in Table 1. The results show substantial performance gains across the evaluated settings. Compared with agents using unimodal Accessibility Tree inputs and with multimodal or end-to-end baselines, Eous equipped with CI4A achieves higher task success rates while substantially shortening interaction sequences. Additional experiments examine transfer across component libraries, deployment in third-party applications, and the contributions of the complete CI4A interface and its individual modules; detailed settings and results appear in Appendices B.3, B.4, and B.5, respectively.

Specifically, our experimental findings are as follows:

▷ **Eous achieves high task success rates across different modalities.** As shown in Table 1, the CI4A protocol significantly extends the operational boundaries of general LLMs within complex GUI environments: 1) *In the unimodal structural perception track.* Eous in structured-input mode achieves an average success rate of 75.3%, substantially outperforming WebArenaBase (26.4%) and AgentOccam in text-only mode (69.2%). This indicates that compared to standard, lossy compressed Accessibility Tree, the semantic view provided by CI4A provides a better context foundation for the decision process, eliminating uncertainty at the planning level. 2) *In the multimodal perception track.* Eous (Vision Mode) achieves an average success rate of 86.3%. It not only surpasses the general-purpose AgentOccam (70.3%) but also outperforms the specialized end-to-end model Seed-1.8 (61.5%), notably without any domain-specific fine-tuning. This demonstrates that CI4A's high-level tool abstraction effectively circumvents the limitations of visual perception and execution in complex scenarios, achieving an execution fidelity akin to What You Intend Is What You Get.

▷ **The hybrid action space achieves an order-of-magnitude improvement in decision-making efficiency.** As shown in Table 1, Eous drastically reduces the average decision sequence length required to complete tasks from a baseline of 10.7 steps to 4.5 steps, representing a relative reduction of 57.5%. This confirms that the synergy between the high-level semantic tools $\Sigma_{\mathcal{T}}$ introduced by CI4A and the comprehensive semantic view $\mathcal{S}$ effectively truncates decision sequences. Consequently, this substantially accelerates task completion, given that action decision-making constitutes the primary time-consuming operation.

4.3 In-depth Analysis

To investigate the underlying mechanisms driving Eous's performance, we conducted an in-depth analysis across two dimensions: action space distribution and robustness against task complexity. Furthermore, Appendix C provides additional overhead analysis as well as successful and failure case studies to further characterize the mechanisms and limitations of CI4A from the perspectives of system efficiency, representative behavioral patterns, and failure boundaries.

▷ **The semantic escalation of interaction paradigms drives the collapse of physical sequences.** As illustrated in Figure 4, we performed a fine-grained statistical analysis of the action sequences generated by agents in successful tasks. The comparison reveals two distinct interaction patterns:

- Baseline agents produce execution trajectories dominated by Atomic Actions (e.g., `click`, `scroll`). This indicates that these models are constrained by the need to simulate human mechanical micro-operations, resulting in excessively lengthy interaction sequences.
- Eous's execution trajectories are dominated by high-level semantic operations, which effectively replace tedious atomic sequences. This confirms that the CI4A protocol successfully promotes a paradigm shift, specifically an Escalation, from mechanical execution to semantic decision-making. The agent evolves into a tool user capable of recognizing component intent. By invoking deterministic primitives such as `setValue` or `navigateTo`, Eous achieves an order-of-magnitude collapse in interaction sequences that originally required multi-step physical interactions.

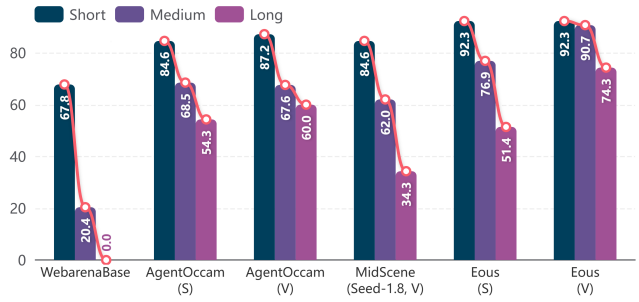

Figure 5: **Performance across task lengths.** Unlike baselines that degrade rapidly in long tasks due to error accumulation, Eous maintains high success rates, demonstrating superior robustness in long-horizon interactions.

▷ **Semantic shortcuts effectively mitigate error accumulation in long-sequence tasks.** The number of interaction steps in a task typically correlates positively with the probability of error. We categorized the test tasks into three difficulty levels based on the standard number of steps required

by the baseline model: Short (1-5 steps), Medium (6-10 steps), and Long (>10 steps). We then compared the success rate variations across groups (as shown in Figure 5):

- Baseline agents exhibit a typical trend of linear decay. As the task sequence lengthens, their average success rates drop precipitously from 81.1% in the Short to 37.15% in the Long. This reflects the classic phenomenon of error accumulation—each additional Click-Observe cycle increases the risk of hallucination or positional drift.
- Eous demonstrates exceptional performance resilience. In Long group tasks, Eous maintains high success rates of 51.4% (structure-based) and 74.3% (vision-based), exhibiting only a marginal decline compared to the Short group. This confirms that CI4A effectively decouples task logic complexity from interaction sequence length. The agent intercepts long-sequence operations, which originally required multiple rounds of trial-and-error exploration, at the decision-making layer, thereby ensuring deterministic performance even in long-horizon, complex tasks.

5 Discussion

Towards a Symbiotic Web Ecosystem for Humans and Agents. Just as the WAI-ARIA standard resolved accessibility challenges via semantic enhancement, CI4A aims to establish an "ARIA for Agents" paradigm. It provides a native, stable, and semantically explicit channel for component-level interaction. We advocate for the adoption of this standard within the frontend ecosystem, enabling Web applications to evolve into environments that are as friendly to agents as they are to humans.

Building on this vision, we discuss the limitations of the current implementation and the key paths toward this future (see Appendix D for further discussion): (1) **Generalizability of Instrumentation.** Our reference implementation, AntDX, currently relies on intrusive instrumentation of the Ant Design framework. Although this effectively validates the CI4A protocol, we acknowledge that manually adapting every UI library is labor-intensive. Future work will explore **automated instrumentation**, using large language models or static analysis to automatically generate CI4A-compliant wrappers for legacy components, thereby reducing the migration costs of existing Web applications. (2) **Scalability to Open-World Scenarios.** Enabling agents to use CI4A on websites built with UI libraries that do not yet support the protocol is a key next step. We envision a skill-based online adaptation mechanism that identifies common components in online Web applications and dynamically enables temporary CI4A adaptation through **real-time JavaScript code injection**. These dynamically generated semantic interfaces will be encapsulated as reusable skills, enabling agents to interact with these websites through CI4A and move beyond closed environments. (3) **Verifiable Safety and Interaction Guardrails.** Compared with opaque simulated clicks, CI4A's explicit semantic contracts enable **runtime parameter validation** and **human-in-the-loop supervision**. Future work will focus on CI4A-based safety middleware that intercepts and validates high-risk agent instructions, such as payments, thereby improving the trustworthiness of the Agentic Web.

6 Conclusion

This paper presents CI4A, a semantic encapsulation protocol designed to bridge the gap between the high-level intents of large language models and low-level Web interactions. By abstracting complex UI components into standardized tool primitives, CI4A effectively addresses the fragility and inefficiency of existing agents' DOM operations. Building on the CI4A protocol, we develop AntDX as a reference implementation and Eous as a hybrid agent. Experimental results demonstrate that CI4A enables Eous to achieve a task success rate of 86.3% in the CI4A-integrated WebArena environment. They further confirm that introducing semantic tools can substantially "collapse" physical interaction chains, bypassing bottlenecks in perception and execution. This work marks a fundamental shift in the Web interaction paradigm from "vision-based simulation" to "contract-based invocation," laying a solid foundation for an Agentic Web that is natively friendly to agents.

Acknowledgments and Disclosure of Funding

This work was supported by the National Key Research and Development Program of China under Grant 2023YFC3305404.

References

[1] Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *arXiv preprint arXiv:2407.13032*, 2024.

[2] ByteDance Seed. Seed1.8 model card: Towards generalized real-world agency. Model card, 2025. URL `https://github.com/ByteDance-Seed/Seed-1.8`.

[3] Chen Chen, Jiawei Shao, Dakuan Lu, Haoyi Hu, Xiangcheng Liu, Hantao Yao, and Wu Liu. Gui-eyes: Tool-augmented perception for visual grounding in gui agents. *arXiv preprint arXiv:2601.09770*, 2026.

[4] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.505. URL `https://aclanthology.org/2024.acl-long.505/`.

[5] James Craig, Michael Cooper, L Pappas, R Schwerdtfeger, and L Seeman. Accessible rich internet applications (wai-aria) 1.0. *W3C Working Draft*, 2009.

[6] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

[7] Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. *arXiv preprint arXiv:2503.09572*, 2025.

[8] Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, et al. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*, 2024.

[9] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

[10] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.371. URL `https://aclanthology.org/2024.acl-long.371/`.

[11] Kaiwen He, Zhiwei Wang, Chenyi Zhuang, and Jinjie Gu. Recon-act: A self-evolving multi-agent browser-use system via web reconnaissance, tool generation, and task execution. *arXiv preprint arXiv:2509.21072*, 2025.

[12] Shiqi He, Yue Cui, Xinyu Ma, Yaliang Li, Bolin Ding, and Mosharaf Chowdhury. Branch-and-browse: Efficient and controllable web exploration with tree-structured reasoning and action memory. *arXiv preprint arXiv:2510.19838*, 2025.

[13] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL `https://aclanthology.org/2024.acl-long.50/`.

[14] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.

[15] Dongjun Lee, Juyong Lee, Kyuyoung Kim, Jihoon Tack, Jinwoo Shin, Yee Whye Teh, and Kimin Lee. Learning to contextualize web pages for enhanced decision making by llm agents. *arXiv preprint arXiv:2503.10689*, 2025.

[16] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pages 51991–52008. Curran Associates, Inc., 2023. doi: 10.52202/075280-2264. URL https://doi.org/10.52202/075280-2264.

[17] Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. Webglm: towards an efficient web-enhanced question answering system with human preferences. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pages 4549–4560, 2023.

[18] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025.

[19] Xing Han Lù, Gaurav Kamath, Marius Mosbach, and Siva Reddy. Build the web for agents, not agents for the web. *arXiv preprint arXiv:2506.10953*, 2025.

[20] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. OmniParser for pure vision based GUI agent. *arXiv preprint arXiv:2408.00203*, 2024. URL https://arxiv.org/abs/2408.00203.

[21] Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao-yong Wei, Shanru Lin, Hui Liu, Philip S Yu, et al. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6140–6150, 2025.

[22] Xinyu Pang, Ruixin Hong, Hongming Zhang, and Changshui Zhang. Assimilation and accommodation: Task-adaptive hierarchical abstraction for solving web tasks. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14000–14014, 2025.

[23] Viraj Prabhu, Yutong Dai, Matthew Fernandez, Jing Gu, Krithika Ramakrishnan, Yanqi Luo, Silvio Savarese, Caiming Xiong, Junnan Li, Zeyuan Chen, et al. Walt: Web agents that learn tools. *arXiv preprint arXiv:2510.01524*, 2025.

[24] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

[25] Sven Schultze, Meike Verena Kietzmann, Nils-Lucas Schönfeld, and Ruth Stock-Homburg. Building the web for agents: A declarative framework for agent-web interaction. *arXiv preprint arXiv:2511.11287*, 2025. URL https://arxiv.org/abs/2511.11287.

[26] Jiazheng Sun, Te Yang, Jiayang Niu, Mingxuan Li, Yongyong Lu, Ruimeng Yang, and Xin Peng. Fairy: Interactive mobile assistant to real-world tasks via LMM-based multi-agent. *arXiv preprint arXiv:2509.20729v1*, 2025. URL https://arxiv.org/abs/2509.20729v1.

[27] Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, et al. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*, 2025.

[28] vueComponent. Ant design vue: An enterprise-class ui components based on ant design and vue. https://github.com/vueComponent/ant-design-vue, n.d. Accessed: 2026-05-06.

[29] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-Agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In *Advances in Neural Information Processing Systems*, volume 37, pages 2686–2710. Curran Associates, Inc., 2024. doi: 10.52202/079017-0088. URL https://doi.org/10.52202/079017-0088.

[30] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

[31] Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*, 2025.

[32] John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pages 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://doi.org/10.52202/079017-1601.

[33] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for LLM-based web agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oWdzUpOlkX.

[34] Yingxuan Yang, Mulei Ma, Yuxuan Huang, Huacan Chai, Chenyu Gong, Haoran Geng, Yuanjian Zhou, Ying Wen, Meng Fang, Muhao Chen, et al. Agentic web: Weaving the next web with ai agents. *arXiv preprint arXiv:2507.21206*, 2025.

[35] Yuhao Yang, Zhen Yang, Zi-Yi Dou, Anh Nguyen, Keen You, Omar Attia, Andrew Szot, Michael Feng, Ram Ramrakhya, Alexander Toshev, et al. Ultracua: A foundation model for computer use agents with hybrid action. *arXiv preprint arXiv:2510.17790*, 2025.

[36] Xinmiao Yu, Liwen Zhang, Xiaocheng Feng, Yong Jiang, Bing Qin, Pengjun Xie, and Jingren Zhou. Webanchor: Anchoring agent planning to stabilize long-horizon web reasoning. *arXiv preprint arXiv:2601.03164*, 2026.

[37] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Large language model-brained GUI agents: A survey. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=xChvYjvXTp.

[38] Jiayuan Zhang, Kaiquan Chen, Zhihao Lu, Enshen Zhou, Qian Yu, and Jing Zhang. Prune4web: Dom tree pruning programming for web agent. *arXiv preprint arXiv:2511.21398*, 2025.

[39] Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23378–23386, 2025.

[40] Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

[41] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://webarena.dev.

[42] Xiao Zhou, Tao Yu, and YiBing Lin. Midscene.js: Your AI operator for web, android, iOS, automation & testing. GitHub repository, 2025. URL https://github.com/web-infra-dev/midscene.

A Experiment Setup Details

A.1 Reconstruction of WebArena with Modern Web Technologies

We performed a semantics-preserving frontend reconstruction of the WebArena web applications using Vue (3.5.27) and Ant Design (4.2.6). This reconstruction was motivated by the following three considerations:

- First, the original WebArena is built on traditional web technology stacks such as PHP and jQuery, which no longer adequately reflect the technical characteristics of modern real-world web environments. According to the 2024 Stack Overflow Developer Survey, contemporary frontend development is dominated by highly componentized frameworks, with React, Angular, and Vue accounting for 39.5%, 17.1%, and 15.4% of usage, respectively. Modern web applications built on these frameworks typically exhibit deeper and more complex DOM tree structures, along with richer component hierarchies, dynamic rendering behaviors, and state interaction logic. As a result, Web Agents operating in modern web environments must handle substantially higher page-understanding complexity and greater action-execution challenges. In contrast, the shallow DOM structures and relatively simple interaction patterns of the original WebArena have gradually diverged from mainstream industrial practice. Continuing to use it as a benchmark may therefore undermine the external validity of experimental findings in real-world settings.
- The CI4A protocol is designed for modern web technology ecosystems and therefore has limited compatibility with the original WebArena, which is implemented using traditional technology stacks. CI4A can be readily implemented within modern web UI frameworks and can also be naturally integrated into applications built with frameworks such as Vue and React. However, the legacy technology stack underlying the original WebArena lacks stable component abstractions and a unified UI semantic layer, preventing us from achieving CI4A compatibility through simple replacement of the underlying UI component library alone. Therefore, a modern frontend reconstruction of WebArena is a necessary prerequisite for integrating CI4A into the benchmark.
- Although designing an entirely new benchmark could circumvent the technical limitations of the original WebArena, doing so would risk introducing unfair comparisons with prior work. WebArena has been widely adopted for evaluating the task completion capabilities of Web Agents, and fully replacing the benchmark would reduce the comparability of our experimental results with existing studies. Therefore, we instead modernized its frontend implementation while preserving the original tasks, page semantics, and business logic, thereby balancing realism with comparability.

During the reconstruction process, we preserved semantic equivalence between the original and reconstructed pages to the greatest extent possible. Specifically, the reconstruction only replaced native UI components with semantically equivalent Ant Design components, while maintaining strict consistency with the original WebArena in terms of page layout, information content, and business functionality. The reconstruction did not alter the visual or logical flow of tasks, nor did it introduce new interaction targets or remove existing action paths.

Furthermore, we conducted an additional set of experiments to verify that the reconstructed WebArena remains substantially equivalent to the original WebArena at the task level. The corresponding results are provided in Appendix B.1.

A.2 Integration of CI4A and WebArena

We first performed invasive instrumentation on the Ant Design library (4.2.6) to implement the CI4A protocol. We refer to the instrumented version as AntDX (the instrumentation procedure is described in detail in Section 3.2). Since AntDX is fully API-compatible with standard Ant Design, we subsequently modified the dependency configuration file (package.json) of the reconstructed WebArena to replace the original Ant Design dependency with AntDX components. No business logic code was modified beyond this change.

This process suggests that the integration cost of CI4A within modern frontend engineering ecosystems primarily depends on whether the underlying UI component library provides native protocol support, rather than on whether application developers must additionally adapt their business logic. For web applications built with modern frameworks and component libraries such as Vue and React, once the underlying UI component library natively supports CI4A, developers can typically obtain CI4A functionality simply by replacing dependencies or upgrading the component library version, without modifying page semantics or business functionality. In contrast, integrating CI4A into legacy information systems is indeed substantially more challenging. However, these challenges fundamentally arise from the inherent complexity of legacy-system modernization itself—including outdated technology stacks, insufficient component abstraction, tightly coupled frontend-backend architectures, and the absence of a unified UI semantic layer—rather than from additional burdens introduced by the CI4A protocol design. CI4A is therefore better positioned as an infrastructural

capability within the modern web ecosystem, where support at the UI-library and framework layers can be transparently propagated upward to application developers.

A.3 Combination of BaseAgent and BaseModel

To evaluate the performance advantages of Eous in complex, industrial-grade Web environments, we selected several representative Web Agent frameworks as baselines and paired them with appropriate foundation models according to their architectural characteristics. Specifically, under the unimodal perception setting, we adopted WebArenaBase and AgentOccam configured in text-only mode as baselines, both powered by GPT-5, to establish a performance benchmark based on Accessibility Tree inputs. Under the multimodal perception setting, we further included the standard configuration of AgentOccam as a vision-enhanced baseline, where both screenshots and Accessibility Trees are used as inputs, again with GPT-5 as the underlying model. In addition, we selected the MidScene framework, which is designed for end-to-end GUI interaction models, and paired it with the natively supported and officially recommended Seed-1.8 model to fully leverage its strengths in visual perception and end-to-end interaction. Collectively, these configurations characterize the performance upper bounds under text-structured input, vision-enhanced input, and end-to-end GUI interaction paradigms, respectively.

To ensure fair and comprehensive comparisons, we also configured Eous in two corresponding variants for strict head-to-head evaluation. The first variant adopts a structured-input setting, where CI4A-provided semantic triples are injected into the Accessibility Tree, with GPT-5 serving as the foundation model. The second variant adopts a visual-input setting, where page perception is performed using SoM-annotated screenshots while CI4A semantic triples are additionally incorporated into the context, again using GPT-5 as the foundation model. This setup enables direct comparison between different agents under either the same foundation model or the same perception modality, thereby allowing us to evaluate whether introducing the CI4A semantic contract yields stable and significant performance gains.

CI4A represents a harness-level engineering enhancement: rather than directly modifying model parameters, it provides agents with clearer and more reliable page semantics and action constraints through a standardized low-level interaction protocol. Consequently, when evaluating protocols of this kind, using weaker models can easily confound task failures with issues such as reasoning hallucinations, insufficient perceptual capability, or unstable instruction following, rather than reflecting the effectiveness of the protocol itself. To minimize interference from model-capability bottlenecks, we uniformly adopted GPT-5 as the primary foundation model in our main experiments. All GPT-5-based experiments were conducted through the OpenAI API with the temperature set to 0 to improve output determinism, and the maximum context window configured to 128k tokens. For all agents, we capped the maximum number of interaction steps per task at 30 (three times the average interaction length); tasks exceeding this limit were considered failures.

For MidScene, we did not adopt GPT-5 as the foundation model because MidScene is inherently a purely vision-driven UI automation framework whose core strengths rely on visual understanding, UI element recognition, and end-to-end action generation. Its official documentation further describes the Doubao Seed vision model as being specifically optimized for visual understanding and UI element recognition tasks. Meanwhile, Seed-1.8 is designed to support generalized real-world agent tasks by integrating perception, reasoning, and action capabilities within a unified model, explicitly including GUI interaction scenarios. Pairing MidScene with Seed-1.8 therefore allows the framework's architectural advantages as an end-to-end GUI Agent system to be more fully realized, while avoiding underestimation of this class of methods due to paradigm mismatch between the model and the framework. Accordingly, we evaluated MidScene under the model configuration most favorable to its design, enabling it to serve as a strong baseline for the end-to-end visual interaction paradigm.

Furthermore, to further evaluate the adaptability and generalization capability of Eous across different foundation models, we conducted additional experiments using models such as Qwen and Gemini; the corresponding results are provided in Appendix B.2. All supplementary experiments were conducted through the official APIs of the respective models, with the temperature consistently set to 0 to maintain deterministic inference behavior. These supplementary experiments are intended to examine whether the CI4A semantic contract remains effective across different model capability boundaries and interface paradigms, rather than being effective only for GPT-5 specifically.

A.4 Environment Details

All experiments were conducted within a unified local browser environment. The experimental machine ran Windows 11 26H1 with Chromium/Chrome version 147.0. To ensure consistent page rendering and visual input conditions across different agents, we fixed the system display scaling factor at 100% and standardized the browser window resolution to 1920×1080. This setup minimizes the influence of variations in window size, coordinate mapping, and page layout.

In terms of hardware, the experimental platform was equipped with an Intel Core i7-12700K processor, 32 GB of memory, and 1 TB of storage. All experiments were executed under identical software, hardware, and display configurations to minimize potential interference from device performance variability, browser rendering discrepancies, and visual coordinate offsets.

B Additional Experiments

B.1 Consistency Verification Before and After WebArena Reconstruction

To verify whether the frontend modernization process altered the task difficulty of WebArena or introduced additional environmental bias, we conducted an additional consistency evaluation between the original and reconstructed versions of WebArena. Importantly, the goal of our reconstruction was not to preserve the underlying HTML/DOM structure exactly, but rather to migrate the original page implementations to a modern frontend technology stack while maintaining consistency in page semantics, business functionality, visual flow, and task logic. Accordingly, the consistency evaluation focuses on two questions: whether the reconstructed environment preserves the same semantic objectives as the original tasks, and whether it systematically makes tasks easier or more difficult for existing agents.

Frontend reconstruction may introduce two opposing categories of potential effects. On the one hand, pages rebuilt using modern UI component libraries generally exhibit cleaner component boundaries and more consistent control styles, which may make certain page elements easier to recognize and thereby simplify some tasks. On the other hand, modern frontend frameworks and component libraries often introduce deeper DOM hierarchies, more complex component nesting, and additional dynamic rendering structures, resulting in deeper and more complex Accessibility Trees. This, in turn, increases the perceptual burden for agents that rely on structured page observations. Therefore, it is not possible to determine a priori, based solely on implementation characteristics, whether the reconstructed version of WebArena systematically favors a particular class of methods; this question must instead be resolved empirically.

To this end, we selected representative baseline agents and evaluated their task success rates on both the original and reconstructed versions of WebArena. The experimental results are reported in Table 2.

Table 2: Consistency validation results between Original WebArena and Reconstructed WebArena.

Framework	Model	Original WebArena	Reconstructed WebArena
WebArenaBase	GPT-5	30.1%	26.4%
AgentOccam(S)	GPT-4-Turbo	45.6%	39.6%
AgentOccam(S)	GPT-5	71.2%	69.2%

The results show that all baselines experienced slight declines in success rate on the reconstructed version of WebArena. Specifically, WebArenaBase + GPT-5 decreased by 3.7 percentage points, AgentOccam(S) + GPT-4-Turbo decreased by 6.0 percentage points, while the strongest structured-input baseline, AgentOccam(S) + GPT-5, decreased by only 2.0 percentage points. These findings suggest that the frontend reconstruction did not artificially reduce task difficulty through cleaner or more standardized page implementations. Moreover, for stronger agent-model combinations, the performance differences before and after reconstruction remained relatively small, indicating that the reconstruction preserved the original task semantics and operational workflow.

This observation is consistent with our expectations for modern frontend environments. Although the reconstructed WebArena remains consistent with the original version in terms of page layout,

information content, and business functionality, the introduction of modern frontend technologies such as Vue and Ant Design makes its underlying DOM and Accessibility Tree structures more representative of the complexity found in real-world industrial web applications. For agents that rely on Accessibility Trees, this added complexity may indeed introduce additional perceptual burden. However, this should not be interpreted as an unfair property of the benchmark; rather, it reflects the genuine challenges that modern web environments impose on Web Agents.

Fairness of the reconstructed environment. A potential concern is that deeper DOM and Accessibility Tree structures depress baseline performance and thereby amplify Eous's relative advantage. Even using the strongest structured-input baseline's original-WebArena result, AgentOccam(S)+GPT-5 at 71.2%, rather than its reconstructed-WebArena result of 69.2%, Eous's 86.3% success rate still represents a 15.1 percentage-point improvement. Thus, even fully compensating for the baseline's 2.0 percentage-point decline does not eliminate the advantage. Moreover, the reconstructed WebArena is not a simplified environment artificially designed to favor Eous. The triplet $\langle \mathcal{S}, \mathcal{T}, \mathcal{M} \rangle$ does not consist of task hints or answer cues added during reconstruction; it comprises standardized component semantic interfaces defined by CI4A. This information neither changes task goals, page content, or business logic nor provides Eous with additional knowledge beyond the semantics of the page itself. It simply exposes, more explicitly and reliably, interaction semantics already implicit in DOM structures, visual layouts, event handlers, and frontend state. This difference does not constitute benchmark unfairness; it is the protocol-level contribution evaluated here: whether CI4A can provide agents with a more stable interaction interface than conventional DOM/Accessibility Tree or screenshot-based inference under modern Web complexity.

B.2 Performance with Different LLMs

To further evaluate the applicability and stability of CI4A across different LLM backbones, we tested Eous(V) with multiple foundation models. For more intuitive comparison, Table 3 also reports the performance of AgentOccam(V) under the corresponding model configurations.

Table 3: Performance of Eous(V) with different LLM backbones on the CI4A-integrated WebArena benchmark.

Framework	Model	SR (%)	Avg. Steps
AgentOccam(V) [33]	GPT-5	70.3	10.4
Eous(V)	GPT-5	**86.3**	4.7
Eous(V)	Qwen3.5-397b-a17b	78.0	4.8
Eous(V)	Qwen3.5-122b-a10b	68.1	4.7
Eous(V)	Gemini3-pro-preview	84.6	4.5
AgentOccam(V) [33]	GPT-4-Turbo	42.2	8.2
Eous(V)	GPT-4-Turbo	61.0	**3.9**

The experimental results show that Eous(V) consistently maintains strong task-completion performance across different LLM backbones, indicating that the component-level semantic information provided by CI4A is not tied to any specific model. Specifically, Eous(V) with GPT-5 achieved the highest success rate of 86.3%, substantially outperforming AgentOccam(V) with the same GPT-5 backbone, which achieved 70.3%. On Gemini3-Pro-Preview, Eous(V) also achieved an 84.6% success rate, approaching its performance under the GPT-5 setting. For the Qwen-series models, Eous(V) still achieved success rates of 78.0% and 68.1%, respectively. Eous(V) also demonstrated clear advantages in interaction efficiency. Compared with the 10.4 average interaction steps required by AgentOccam(V)+GPT-5, Eous(V)+GPT-5 completed tasks in only 4.7 steps on average. Under the GPT-4-Turbo setting, the average interaction length of Eous(V) was similarly reduced from 8.2 steps for AgentOccam(V) to 3.9 steps.

The results across different LLM backbones further demonstrate that CI4A remains effective under varying model capability boundaries. The performance gains achieved by Eous are therefore not attributable to GPT-5-specific capabilities, but instead stem from the structured semantic interaction mechanism introduced by the CI4A protocol. Stronger models can further amplify the benefits of CI4A; however, even under alternative model settings such as GPT-4-Turbo and Qwen, Eous consistently exhibits stable improvements in both task success rate and interaction efficiency.

B.3 Experiments on Different Component Libraries

To examine generality across component libraries and frontend frameworks, we instrumented Material UI (React) and Element Plus (Vue 3) in addition to AntDX, and reconstructed corresponding versions of WebArena. The three environments preserve the same business logic, evaluation oracle, and initial state. Following the main experimental settings, we evaluated Eous(V) and AgentOccam(V), both with GPT-5. Table 4 reports the results.

Table 4: Performance across component libraries in CI4A-integrated WebArena environments. Eous maintains higher success rates and shorter interaction sequences.

Component library	Agent	SR (%)	Avg. steps
AntDX (Ant Design Vue)	Eous(V)	86.3	4.7
	AgentOccam(V)	70.3	10.4
MaterialX (Material UI)	Eous(V)	85.2	5.5
	AgentOccam(V)	68.7	9.8
ElementX (Element Plus)	Eous(V)	86.8	5.4
	AgentOccam(V)	69.8	11.9

Both agents exhibit broadly consistent performance across the three libraries. Eous improves success rates over AgentOccam by at least 16 percentage points and reduces interaction steps by approximately half. Across the three libraries, the gains remain broadly consistent despite changes to the underlying component library. These results support both the stability and transferability of CI4A's performance benefits and its ability to generalize across component libraries. The adaptation costs of all three implementations are reported in Appendix C.1.

B.4 Real-World Deployment Case Studies

To examine deployment in existing applications, we selected two widely followed open-source projects on GitHub: antd-vue-admin, an administration template based on ant-design-vue, and traccar-web, an industrial-grade vehicle-tracking platform based on Material UI. We replaced their component dependencies with AntDX and MaterialX at revisions `64c03db` and `f702dc0`, respectively. Integration required only changes to dependency declarations in `package.json`, with no application-code modifications; both projects built and ran successfully. After establishing logged-in sessions, we designed ten representative tasks for each project and evaluated Eous(V)+GPT-5 and AgentOccam(V)+GPT-5.

Table 5: Results in third-party applications. Eous completes more tasks with fewer average interaction steps.

Environment	Agent	Successes	Avg. steps
antd-vue-admin (AntDX)	Eous(V)	10/10	5.3
	AgentOccam(V)	8/10	10.0
traccar-web (MaterialX)	Eous(V)	7/10	7.5
	AgentOccam(V)	5/10	14.2

As shown in Table 5, Eous completes more tasks and requires fewer interaction steps in both projects. These case studies demonstrate the feasibility of acquiring CI4A capabilities through dependency replacement in existing third-party applications, supporting the protocol's engineering transferability.

B.5 Ablation Study

To distinguish the contribution of CI4A from the Eous agent configuration, we first introduce an Eous-Atomic control. This variant retains the full Eous Planner–Decider–Reflector architecture and main prompt templates, but disables CI4A's semantic state, component tools, and interaction metadata. It executes tasks through atomic operations such as click, type, and scroll.

To assess the contribution of each component within the semantic triplet $\langle \mathcal{S}, \mathcal{T}, \mathcal{M} \rangle$ in the CI4A protocol, we performed a leave-one-out ablation study. Specifically, while keeping the model, task

set, browser environment, and interaction budget fixed, we removed one module from the triplet at a time and measured the resulting changes in task success rate and average steps.

Table 6: CI4A ablation results in the integrated WebArena environment, including Eous-Atomic and leave-one-out variants. Removing CI4A or any of its modules reduces task success.

Framework	SR (%)	Avg. Steps
Eous(V)	**86.3**	4.7
Eous-Atomic	69.2	8.5
Eous(V) w/o $\mathcal{S}$	76.9	8.4
Eous(V) w/o $\mathcal{T}$	74.7	7.3
Eous(V) w/o $\mathcal{M}$	83.0	**4.6**

As shown in Table 6, full Eous improves success over Eous-Atomic by 17.1 percentage points and reduces average steps from 8.5 to 4.7, characterizing the gain from the complete CI4A interface with the agent configuration held fixed. The leave-one-out results show that removing any one of the modules leads to a decline in task success rate, indicating that the three components contribute to agent task performance through complementary mechanisms related to semantic understanding, action execution, and constraint validation, respectively.

- First, the component-level contextual information provided by the semantic state view $\mathcal{S}$ serves as a critical foundation for both decision quality and execution efficiency. Removing $\mathcal{S}$ reduces the success rate of Eous(V) from 86.3% to 76.9%, while increasing the average interaction length from 4.7 to 8.4 steps. This suggests that, without component-level semantic context, the agent must rely on additional low-level interactions to infer page state and discover hidden information. As a result, decision uncertainty increases substantially, making the model more prone to hallucinations. For example, in certain tasks, the agent can still access the *navigateTo* tool associated with a Menu component. However, without understanding the semantic role of the component or the meaning of the target page, the LLM may construct invalid navigation targets. After navigation failure, the agent must revert to atomic operations such as scrolling and clicking to re-explore the page, thereby reducing task success rate and increasing interaction length.
- Second, the primary role of the executable toolset $\mathcal{T}$ is to encapsulate multiple low-level atomic actions into semantically explicit high-level tool invocations, thereby shortening action chains and reducing the risk of cumulative errors. Removing $\mathcal{T}$ decreases the success rate of Eous(V) to 74.7% and increases the average interaction length to 7.3 steps. Under this setting, the agent can still access the semantic state view $\mathcal{S}$ and therefore retains an understanding of the page structure and target component semantics. However, it can no longer directly invoke component-level tools and must instead rely on low-level atomic actions such as clicking, typing, and scrolling. Consequently, interactions that could previously be completed through a single high-level tool invocation are decomposed into multiple low-level operations, increasing the likelihood of coordinate localization errors, state-synchronization failures, and intermediate-step breakdowns. Nevertheless, because the semantic view remains available, the agent can still avoid substantial amounts of ineffective scrolling and blind navigation during exploration. As a result, its average interaction length remains lower than in the setting where $\mathcal{S}$ is removed.
- Finally, interaction metadata $\mathcal{M}$ mainly improves action validity by providing business constraints, input requirements, and operation preconditions. After removing $\mathcal{M}$, the success rate of Eous(V) decreased to 83.0%, but the average number of interaction steps remained largely unchanged. This result indicates that $\mathcal{M}$ has relatively limited influence on task efficiency, but contributes to final task success. Without $\mathcal{M}$, the agent can still locate target components and perform the corresponding operations. Its average number of interaction steps therefore remains largely unchanged. However, due to the lack of explicit prompts regarding operation constraints, such as required form fields, input format requirements, submission conditions, or component disabled states, the agent is more likely to produce seemingly reasonable but actually invalid operations, resulting in submission failures or inability to complete business processes.

Overall, $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{M}$ serve distinct yet complementary roles: $\mathcal{S}$ provides the semantic foundation for page understanding, $\mathcal{T}$ provides high-level interfaces for action execution, and $\mathcal{M}$ provides the constraint information necessary to ensure action validity. The complete CI4A semantic triplet

therefore improves both task success rate and execution efficiency, whereas removing any individual module degrades the performance of Eous in complex web environments.

C Additional Analysis

C.1 Cost Analysis

To evaluate the practical deployability of the CI4A protocol, we analyze it from three perspectives: browser runtime overhead, model inference cost, and engineering integration cost.

▷ **Browser Main-Thread Overhead.** In our reference implementation, AntDX, the additional overhead introduced by CI4A on the browser main thread is minimal. During component mounting, the global registration process merely inserts an in-memory reference to the component instance into the *window.__ci4a__* hash table, without involving complex computation, DOM traversal, or synchronous blocking operations. The semantic state view $\mathcal{S}$ is likewise not generated continuously during page rendering, but instead constructed on demand only when the agent explicitly invokes a CI4A interface call. Consequently, even for complex pages containing thousands of DOM nodes, the lightweight proxy mechanism introduced by AntDX incurs only sub-millisecond main-thread overhead and does not block normal page rendering, event handling, or user interaction.

▷ **Model Inference Cost.** From the perspective of a single request, injecting the semantic triplet $\langle \mathcal{S}, \mathcal{T}, \mathcal{M} \rangle$ into the context does indeed increase prompt length to some extent. However, when considering the end-to-end task execution process, CI4A instead significantly reduces overall token consumption and system latency. Traditional perception paradigms based on DOM structures, Accessibility Trees, or visual screenshots typically require multi-round exploration, where agents progressively infer page states and executable actions through scrolling, clicking, backtracking, and repeated observation. In contrast, by explicitly exposing component states, executable tools, and interaction constraints, CI4A enables agents to complete tasks in substantially fewer steps. As shown in Table 1, Eous reduces the average interaction length from 10.7 to 4.5 steps, directly reducing the number of LLM calls, generation waiting time, and token accumulation from multi-turn histories by more than half. Therefore, although each individual request contains additional structured semantic information, the reductions in interaction length produce sufficient savings in both inference cost and execution latency to offset the overhead introduced by a single semantic-state injection. Furthermore, in terms of actual token usage, Eous consumes approximately 68k tokens per task on average, whereas AgentOccam consumes approximately 71k tokens per task. These results further indicate that CI4A does not merely trade longer prompts for higher success rates; instead, it reduces overall task-level inference cost by minimizing ineffective exploration and shortening interaction chains.

▷ **Engineering Cost of Instrumentation.** Integrating CI4A requires upfront instrumentation of the underlying UI component library. Table 7 summarizes its scale and cost across the three implementations. This cost is incurred once at the reusable library level: applications built on an instrumented library can inherit CI4A capabilities by replacing or upgrading dependencies without modifying business logic, as illustrated by the two deployments in Appendix B.4. The primary engineering burden therefore falls on component-library maintainers or infrastructure providers, rather than on each application developer.

Table 7: Instrumentation scale and engineering cost. Net added lines of code (LOC) are obtained from Git diffs; person-days are manually recorded.

Component library	Components	Net added LOC	Person-days
AntDX	23	3,998	—
MaterialX	15	1,095	≈ 2.2
ElementX	24	1,165	≈ 1.8

AntDX was developed alongside the protocol and included interface exploration, refactoring, and compatibility adjustments; its actual development time can no longer be reconstructed reliably. MaterialX and ElementX were developed after the protocol and reference implementation stabilized, by human developers assisted by Claude Code. Their person-days therefore mainly reflect the marginal cost of adapting another component library. Differences in component coverage and internal architecture preclude a strict comparison of development efficiency.

C.2 Successful Case Studies

To analyze how the findings in Section 4.3 translate to real-world scenarios, we selected two representative complex components Cascader and DatePicker for detailed case studies.

▷ **Case 1: Eliminating Information Asymmetry with Cascader.** When interacting with the Cascader component, Eous requires an average of only 1.1 steps, representing a relative reduction of 73.8% compared to the baseline's 4.2 steps; meanwhile, the accuracy rose from the baseline's 74% to 100%. Constrained by the Progressive Disclosure mechanism inherent to the physical interface, baseline agents handling multi-level linkage (e.g., Province / City / District) must execute a lengthy sequence: Click Expand → Visual Scan/Scroll → Click Target. Crucially, they can only perceive the next level of options after expansion. This blind man feeling an elephant style of interaction incurs extremely high exploration costs. In contrast, benefiting from the Semantic State View $\mathcal{S}$ injected by CI4A, Eous possesses semantic precognition. It acquires all candidate values within the component upon the initial observation without requiring any prerequisite interactions (e.g., `{name: "cascader", status: [{value: "westlake", path: ["Zhejiang", "Hangzhou", "West Lake"]}, ...]}`). This enables Eous to bypass the exploratory phase and directly plan the instruction `call("cascader", "setValue", "westlake")`, achieving a complexity leap from $O(N)$ to $O(1)$.

▷ **Case 2: Bypassing Physical Constraints and Environmental Interference with DatePicker.** Eous requires only 1.0 steps on average when interacting with the DatePicker component—a relative reduction of 74.4% compared to the baseline's 3.9 steps. Baseline agents face rigorous micro-operation challenges during cross-month/year date selection tasks, requiring continuous pagination clicks, grid coordinate identification, and avoidance of disabled date interference. A single click deviation fractures the entire task chain. We observed two typical failure modes: 1) *Format Hallucination*: AgentOccam attempted to bypass the complex date panel by typing text directly into the input field. However, absent clear data constraints, AgentOccam was rejected by frontend validation logic due to its inability to infer the strict input format (e.g., inputting *`2025/12/31`* when the component required *`YYYY-MM-DD`*). This format hallucination resulted in invalid operations. 2) *Environmental Fragility*: The MidScene-based pure vision model exhibited extreme sensitivity to the rendering environment. Minor discrepancies in the display configuration of the experimental device caused a systematic drift in screen coordinates, rendering all date selection clicks ineffective during a specific test run. Although this outlier result was discarded, the phenomenon profoundly highlights the robustness defects inherent in pure pixel interaction when facing environmental heterogeneity. In contrast, Eous utilizes the *`setValue`* tool to establish a semantic shortcut. By bypassing low-level UI rendering logic and directly injecting a target value that satisfies the type constraints in the interaction metadata (via `call("datepicker", "setValue", "2025-12-31")`), this mechanism circumvents the uncertainty of the physical rendering layer, ensuring absolute precision in execution.

C.3 Failure Case Studies

We manually annotated Eous's failed tasks using five categories: low-level execution errors (a broadly reasonable plan but an incorrect interaction, field, or navigation target); high-level reasoning or planning errors (misunderstanding task goals, business rules, or operational workflows, preventing the agent from forming a valid plan or causing reasoning errors during execution); hallucinations (judgments inconsistent with page state or tool interfaces, such as misidentifying page content, constructing nonexistent tool calls, or generating operation parameters in an invalid format); timeouts (exceeding the interaction budget before completion); and other failures. Table 8 reports the counts.

Table 8: Manually annotated failure categories for Eous.

Agent	Low-level execution	Reasoning/ planning	Hallucination	Timeout	Other
Eous(V)	11	7	0	5	2
Eous(S)	17	18	1	7	2

Among Eous(V)'s failures, low-level execution errors account for 11 cases (44%), high-level reasoning or planning errors for 7 (28%), timeouts for 5 (20%), and other errors for 2 (8%). We observed no

failures directly attributable to evident hallucinations in this sample, suggesting that CI4A's semantic constraints help suppress tool misuse and page-content or format misinterpretation.

Low-level execution errors include misunderstanding business interfaces and selecting incorrect pages. For example, when a task requires increasing a product's price by 15, an update form expects the final price, but the agent may enter the increment itself. The agent may also navigate to a functionally similar but incorrect page and mistakenly conclude that it can complete the task there. High-level failures concern implicit workflows: when asked to add a SKU with specified sizes and colors to an existing product, the agent may attempt to create a new product instead of navigating to the existing product's specification controls. These failures reflect misunderstanding of business-object relationships and operational preconditions, rather than missing tools or invisible interface elements.

The failure distribution of Eous(S) is broadly consistent with that of Eous(V), with failures still primarily arising from low-level execution errors and high-level reasoning or planning failures. However, Eous(S) exhibits a larger overall number of failures, along with a slightly higher proportion of high-level reasoning and planning errors. This suggests that, in the absence of visual context, the structured-input setting has relatively limited capacity to understand global page layouts and business workflows.

Overall, the failure-case analysis indicates that most Eous failures do not stem from interface limitations of the CI4A protocol itself, but rather from the limited ability of current large language models to reason about complex business logic, implicit operational workflows, and long-horizon task planning. Even though CI4A already provides relatively stable component-level states, executable actions, and operational constraints, models may still select incorrect paths or misinterpret task objectives within complex business flows. These findings suggest that CI4A has substantially alleviated the traditional "perception-to-execution" bottleneck of Web Agents. However, long-horizon business reasoning remains a central unresolved challenge for current large language models operating in complex web-task environments.

D Further Discussion

For CI4A to be effective across a broader range of Web environments, continued progress is needed in adaptation efficiency, generalization across environments, and interaction safety.

CI4A has now been implemented in three component libraries—AntDX, MaterialX, and ElementX—and application integration has been examined through two third-party projects (Appendices B.3 and B.4). These results extend validation of the protocol across the React and Vue ecosystems, but the implementations still depend on component-library instrumentation: if every UI component library must be adapted manually, the engineering cost will scale linearly with ecosystem size, making it difficult to directly support the highly heterogeneous frontend implementations found across the open web. Nevertheless, modern frontend development has widely adopted component-based frameworks such as React and Vue, where component boundaries, state management, and event-handling logic typically follow relatively stable engineering patterns. This creates opportunities for automated instrumentation. Future work could explore automated adaptation approaches based on static analysis, runtime probing, and LLM-based code generation to automatically identify component states, event interfaces, and business constraints, and to generate CI4A-compliant semantic wrappers for existing components. Through such approaches, the integration cost of CI4A could potentially be reduced from manual per-component adaptation to a semi-automated or fully automated framework-level migration process, thereby improving its deployability across existing web applications.

Our real-world deployment case studies (Appendix B.4) validate CI4A in real-world settings, but many websites on the open Web still lack native CI4A support, making it difficult for agents to directly obtain standardized component-level semantic interfaces. Consequently, a key future direction is enabling agents to interact through CI4A even on websites without native support. **We envision introducing an On-the-fly CI4A Adaptation mechanism.** Specifically, when accessing online Web applications, agents could automatically identify common component patterns on the page, such as menus, tables, forms, filters, paginators, and modal dialogs, and dynamically construct temporary CI4A wrappers through runtime JavaScript injection. These dynamically generated semantic interfaces could then be further abstracted into reusable skills, enabling agents to reuse existing adaptation capabilities when subsequently accessing structurally similar websites. Through this approach, CI4A could evolve from a "native protocol for instrumented component libraries" into a

"dynamically adaptable interaction abstraction for the open Web," thereby overcoming environmental limitations.

CI4A does not expose low-level system APIs, nor does it grant agents capabilities beyond the original permission boundaries of the webpage. It semantically encapsulates legitimate component capabilities under the current user session. State interfaces expose only whitelisted information already delivered to the client. Tool invocations must be equivalent to legitimate user-operation sequences in application-level outcomes and side effects, preserving existing event handlers, form validation, business confirmations, and backend authorization. The global registrar accepts only registered component tools, validates component identifiers, tool names, and parameter schemas, and neither accepts nor executes arbitrary JavaScript. Consequently, from a permission-boundary perspective, CI4A does not bypass the original access-control policies or business constraints of web applications. However, in scenarios involving sensitive data or high-risk operations—such as payments, permission modifications, account settings, or privacy-related workflows—agents may still trigger actions that do not align with the user's actual intent due to task-understanding errors, contextual misinterpretation, or malicious prompt injection. These risks are not unique to CI4A, but are instead inherent challenges faced by all Agentic Web interaction systems. Compared with opaque mouse clicks or coordinate-based operations, the key advantage of CI4A lies in the explicit semantic structure of its interaction process: every action can be represented as a structured invocation containing component states, action types, parameters, and metadata constraints. This property enables runtime parameter validation, risk classification, human-in-the-loop supervision, and operational auditing. Future work will further explore CI4A-based security middleware capable of intercepting, validating, and confirming agent instructions before high-risk actions are executed. For example, for sensitive operations such as payments, deletion, or permission changes, the system could require additional constraint checks, user confirmation, or policy verification, thereby improving the trustworthiness and controllability of the Agentic Web while preserving agent automation capabilities.

Overall, CI4A still faces challenges related to component-library adaptation costs and generalization across the open web. However, these challenges do not diminish its core value as a "web semantic interface for intelligent agents." Instead, they highlight several critical directions that must be addressed as the Agentic Web ecosystem moves toward large-scale deployment: expanding from a small number of instrumented component libraries to diverse frontend ecosystems, extending from controlled evaluation environments to open-web scenarios, and ultimately establishing reusable, auditable, and governable infrastructure for agent interaction. As an initial step toward a web ecosystem in which humans and intelligent agents can coexist and collaborate, CI4A introduces a new organizational paradigm for interactions between web applications and agents, while also laying the groundwork for future agent-friendly web infrastructure.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes. We claimed in Abstract and Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. We discussed this in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper do not have theoretical part.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes. We provided all the details in Appendices A and B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We submitted the code in accordance with the NeurIPS code and data submission guidelines.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes. We listed all the details in Section 4 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our experiments take too much computational resources so that we cannot run it too many times to get the error bar.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. We discussed this in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe we followed the code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes. We discussed this in Appendix D.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our method is not designed for generation.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes. We discussed this in Appendix A.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The development of the core methods in this research does not involve LLMs as any important, original, or non-standard component. LLMs were used only for writing assistance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.