See, Think, Act: Teaching Multimodal Agents to Effectively Interact with GUI by Identifying Toggles

Supplementary Material

Contents

A Detailed Experimental Setup **1**
- A.1. Details of State Control Benchmark 1
- A.2. Evaluation of Action Match Rate 3
- A.3. Details of Agentic Benchmark 4
- A.4. Details of Dynamic Evaluation Benchmark . 5
- A.5. Implementation Details of StaR 5

B Additional Results and Analyses **6**
- B.1. Detailed Evaluation Results of Existing Multimodal Agents on State Control Benchmark 6
- B.2. Detailed Performance on Agentic Benchmarks 8
- B.3. Ablation Studies on StaR components 9
- B.4. Case Studies 9

C Prompts **9**

A. Detailed Experimental Setup

This section provides the comprehensive experimental configuration. Section A.1 presents details for the state control benchmark. Section A.2 outlines the detailed evaluation process for AMR. Section A.3 presents details for the agentic benchmark. Section A.4 presents the details for the dynamic evaluation Benchmark. Section A.5 provides the training and testing implementation details of StaR.

A.1. Details of State Control Benchmark

We present the details of the data sources, the three-step annotation pipeline, data quality, and the evaluation metrics for the state control benchmark as follows.

► **Data Source.** We construct the state control benchmark from the public agentic datasets, including AMEX [2], RICOSCA [7], GUIAct-Mobile [3], AndroidWorld [13], AITW [12], and the grounding dataset of OS-Atlas [16]. These datasets cover a wide range of mobile applications and screen resolutions on the mobile platform, with abundant interfaces that contain GUI toggles and toggle control instructions. We filter the screenshots corresponding to toggle control instructions that include keywords related to toggle state control, such as "turn on/off". "enable/disable", from these datasets for subsequent annotation.

► **Details of the Three-step Annotation Pipeline.** Building a high-quality toggle state control benchmark requires precise annotation of toggle position, toggle state, and toggle functionality. We decompose the complex annotation process into a three-step annotation pipeline. The implementation details are presented as follows.

(i) **Widget Parsing.** Given that screenshots $s \in \mathbb{S}$ corresponding to toggle control instructions $u_t \in \mathbb{U}$ may contain more than one GUI toggle (as shown in Figure 1), we apply OminiParser [8] to parse additional bounding boxes $b_p \in \mathbb{B}$ for clickable elements from these screenshots to enrich toggle diversity. We then merge the original and parsed bounding boxes into a unified set $\{b\} = \{b_o\} \cup \{b_p\}$, which serves as the basis for subsequent toggle identification.

(ii) **Toggle Identification.** Since the bounding boxes from the previous step may not always correspond to GUI toggles, we filter out non-toggle bounding boxes. Specifically, we adopt proprietary MLLMs, Qwen-2-VL-72B [15] (denoted as $\mathcal{Q}$, **the best zero-shot baseline in Section 3.2**) and GLM-4V-Flash [5] (denoted as $\mathcal{G}$, **freely available**), as independent annotators to recognize GUI toggles. Since proprietary MLLMs perform poorly in GUI grounding (as shown in Section 3.2), directly identifying toggles from bounding boxes is both challenging and inaccurate. Therefore, for each bounding box b and its associated screenshot s, we highlight b on s to obtain box-highlighted screenshot s_b, which visually guides proprietary MLLMs to focus on the widget bounded by b for more accurate widget recognition. Each annotator ($\mathcal{G}$ and $\mathcal{Q}$) serves as an indicator $\mathcal{I}$ to determine whether b is a toggle based on box-highlighted screenshot s_b. The prompt template is provided in Appendix C. To ensure reliability, we apply inter-annotator agreement: only when both $\mathcal{G}$ and $\mathcal{Q}$ classify b as a toggle do we retain $\langle s_b, b \rangle$. This process is formally defined as follows, where $\mathbf{1}[\cdot]$ is the indicator function that outputs 1 when the inner condition is met, otherwise outputs 0.

$$\begin{aligned} &\mathcal{I}_{\mathcal{G}}(s_b, b),\ \mathcal{I}_{\mathcal{Q}}(s_b, b) \in \{0, 1\}, \\ &\mathcal{I}_m(s_b, b) = m(s_b, b),\ m \in \{\mathcal{G}, \mathcal{Q}\}, \\ &\mathcal{I}(s_b, b) = \mathbf{1}\left[\mathcal{I}_{\mathcal{G}}(s_b, b) = \mathcal{I}_{\mathcal{Q}}(s_b, b)\right]. \end{aligned} \tag{1}$$

(iii) **State-functionality Annotation.** This step employs $\mathcal{G}$ and $\mathcal{Q}$ as independent annotators to label the GUI toggle state and its functionality. Given the bounding box b of a GUI toggle and the corresponding box-highlighted screenshot s_b, each annotator independently determines the current toggle state σ, where 0 indicates the toggle is currently off and 1 indicates it is on, and toggle functionality f. The prompt template for this annotation is provided in Appendix C. To ensure label reliability, we apply inter-annotator agreement: only when both $\mathcal{G}$ and $\mathcal{Q}$ produce identical annotations for both σ and f, we accept the final

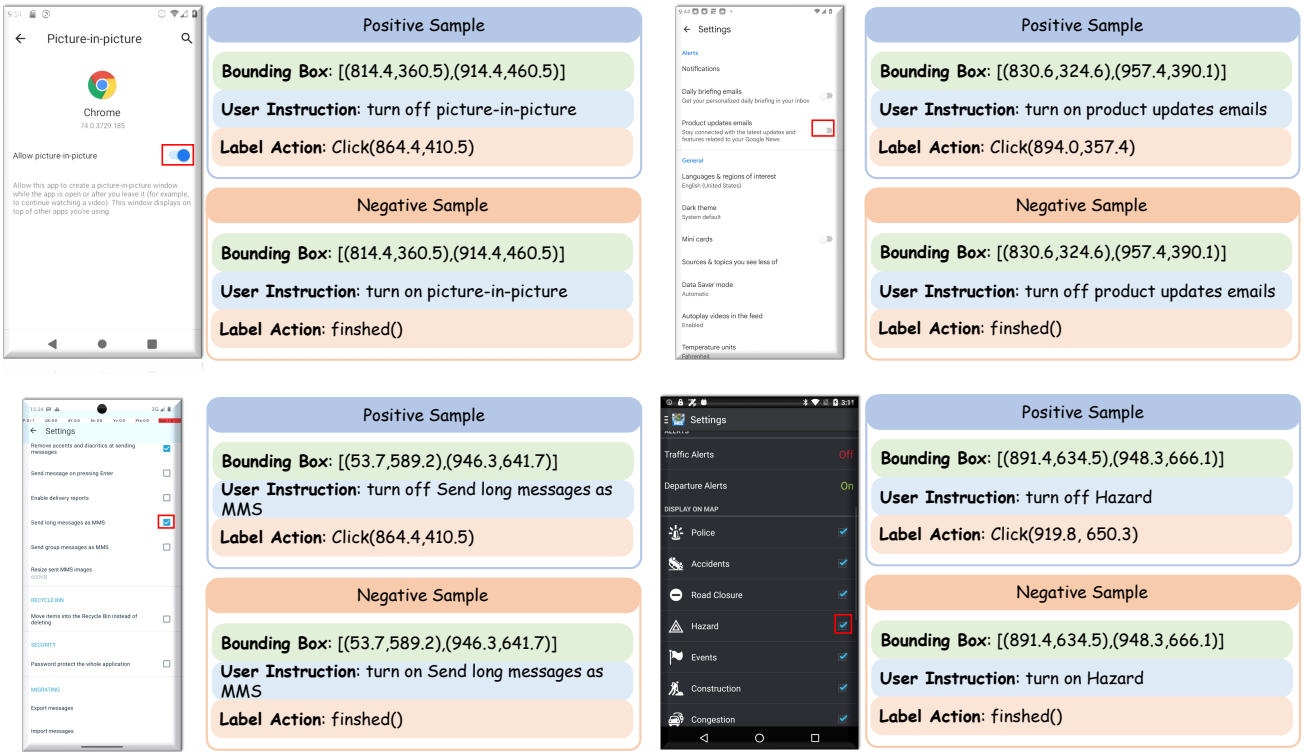

Figure 1. Examples from the test split of state control benchmark, with target toggles highlighted in red boxes for clarity.

Table 1. Data source distribution for the state control benchmark.

Split	AITW [12]	RICOSCA [7]	OS-Atlas [16]	AMEX [2]	AndroidWorld [13]	GUIAct-Mobile [3]	Total
Train	68,380	4,144	496	444	130	58	73,652
Test	7,558	496	60	56	12	2	8,184

annotation $\langle s_b, b, \sigma, f \rangle$. The process of state-functionality annotation is formally represented as follows.

$$\begin{aligned} &\sigma_m(s_b, b),\ f_m(s_b, b) = m(s_b, b),\ m \in \{\mathcal{G}, \mathcal{Q}\}, \\ &\mathcal{I}_\sigma(s_b, b) = \mathbf{1}\left[\sigma_\mathcal{G}(s_b, b) = \sigma_\mathcal{Q}(s_b, b)\right], \\ &\mathcal{I}_f(s_b, b) = \mathbf{1}\left[f_\mathcal{G}(s_b, b) = f_\mathcal{Q}(s_b, b)\right]. \end{aligned} \tag{2}$$

After annotation, we obtain 40,918 screenshots. For more comprehensive and practical evaluation, we replace the box-highlighted screenshots s_b with the original screenshots s and then expand each record with both positive and negative instructions based on the annotated toggle. Positive instructions u_p require clicking the toggle to change the state, while negative ones u_n require maintaining the current state. This yields 81,836 samples, which are then split into 73,652 balanced training and 8,184 testing samples, where each positive sample corresponds to a negative sample within the same split. The data source distribution for the state control benchmark is shown in Table 1.

Examples of test samples are provided in Figure 1, with target toggles highlighted in red boxes for clarity.

► **Data Quality.** Although proprietary MLLMs are not fully reliable for precise annotation, we apply two strategies to ensure the quality of the state control benchmark. First, we highlight the bounding box of each GUI toggle on the corresponding screenshot. This helps proprietary MLLMs focus on the widget for more accurate recognition, mitigating the limitation of low grounding ability. Second, we employ inter-annotator agreement to filter out inconsistent annotations. This strategy improves the reliability of the annotation. The details are as follows.

Assume two proprietary MLLMs $\mathcal{Q}$ and $\mathcal{G}$ act as independent annotators with probabilities p_1 and p_2 of correct annotation. Since we retain only matching annotations of the two annotators, the proportion p of correct annotations among all retained annotations is given by:

$$p = \frac{p_1 p_2}{p_1 p_2 + (1 - p_1)(1 - p_2)}. \tag{3}$$

As we highlight the bounding box of each GUI toggle to reduce grounding errors and improve recognition, p_1 and p_2 are greatly improved. We assume $p_1 > p_2$, then:

Table 2. Estimated probabilities of correct annotation p_1 and p_2 for two proprietary MLLMs, obtained by sampling 100 instances of state-functionality annotation and manually verifying their quality.

p_1	p_2	p (Theoretical estimation)
0.81	0.80	0.946

$$p - p_1 = \frac{p_1(1-p_1)(2p_2-1)}{p_1p_2 + (1-p_1)(1-p_2)}. \tag{4}$$

The condition $p > p_1$ holds when $p_2 > 0.5$, indicating that in this case, p exceeds both p_1 and p_2. We estimate p_1 and p_2 by sampling 100 instances of state-functionality annotation and manually checking the annotation quality, with results presented in Table 2. Since $p_2 > 0.5$, the retained annotations through inter-annotator agreement are more reliable than those from a single proprietary MLLM.

To assess the final benchmark quality, we manually verify 200 randomly sampled disjoint instances, finding that **92.5% of functionality and 91% of state annotations** match the ground truth, which aligns with our estimation in Table 2. This confirms that our annotation pipeline ensures high annotation accuracy and overall benchmark reliability.

► **Metrics.** The comprehensive metric definitions for the state control benchmark are provided as follows. Let $\hat{t}_i$ and t_i denote the predicted and ground truth action type of i-th sample, respectively. Let $\hat{a}_i$, a_i, a_i^f represent the predicted action, ground truth action, and corresponding inverse toggle action, respectively. $\mathbf{1}[\cdot]$ denotes the indicator function. $\mathcal{P}$ and $\mathcal{N}$ denote the sets of positive and negative samples, respectively, and N denotes the total number of samples.

(i) **Overall Type Match Rate (O-TMR)** ↑: Proportion of test samples where the predicted action type (`CLICK` or `COMPLETED`) matches the ground truth. The formal definition of O-TMR is provided in Equation 5.

$$\text{O-TMR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{t}_i = t_i\right]. \tag{5}$$

(ii) **Overall Action Match Rate (O-AMR)** ↑: Proportion of test samples where the predicted action exactly matches the ground truth, considering both action type and click coordinate accuracy. O-AMR is the most critical metric on state control benchmark, reflecting the overall action precision of multimodal agents. The detailed evaluation process of action match rate is provided in Appendix A.2. The formal definition of O-AMR is provided in Equation 6.

$$\text{O-AMR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{a}_i = a_i\right]. \tag{6}$$

(iii) **Positive Type Match Rate (P-TMR)** ↑: Proportion of positive samples where the predicted action type (`CLICK`) matches the ground truth. The formal definition of P-TMR is provided in Equation 7.

$$\text{P-TMR} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \mathbf{1}\left[\hat{t}_i = t_i\right]. \tag{7}$$

(iv) **Positive Action Match Rate (P-AMR)** ↑: Proportion of positive samples where the predicted action exactly matches the ground truth, considering both type and click coordinate accuracy. The formal definition of P-AMR is provided in Equation 8.

$$\text{P-AMR} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \mathbf{1}\left[\hat{a}_i = a_i\right]. \tag{8}$$

(v) **Positive False Negative Rate (P-FNR)** ↓: Proportion of positive samples incorrectly predicted as negative (`COMPLETED`), reflecting the severity of false negatives. The formal definition of P-FNR is provided in Equation 9.

$$\text{P-FNR} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \mathbf{1}\left[\hat{t}_i = \texttt{COMPLETED}\right]. \tag{9}$$

(vi) **Negative Action Match Rate (N-AMR)** ↑: Proportion of negative samples where the predicted action (`COMPLETED`) matches the ground truth. The formal definition of N-AMR is provided in Equation 10.

$$\text{N-AMR} = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} \mathbf{1}\left[\hat{a}_i = a_i\right]. \tag{10}$$

(vii) **Negative False Positive Type Rate (N-FPTR)** ↓: Proportion of negative samples incorrectly predicted as `CLICK`, reflecting the false-positive tendency. The formal definition of N-FPTR is provided in Equation 11.

$$\text{N-FPTR} = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} \mathbf{1}\left[\hat{t}_i = \texttt{CLICK}\right]. \tag{11}$$

(viii) **Negative False Positive Rate (N-FPR)** ↓: Proportion of negative samples where the predicted `CLICK` coincides with the corresponding positive action, indicating the severity of false positives. The formal definition of N-FPR is provided in Equation 12.

$$\text{N-FPR} = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} \mathbf{1}\left[\hat{a}_i = a_i^f \ \wedge\ \hat{t}_i = \texttt{CLICK}\right]. \tag{12}$$

A.2. Evaluation of Action Match Rate

The exact action match rate (AMR) is a key metric for evaluating step-wise action prediction, as it requires both the action type t and parameters p (e.g., coordinates, app name, text input) to match the ground truth. The action space, along with corresponding parameters and descriptions in

Table 3. The action space with corresponding action parameters and descriptions in our experiments

Action Type	Action Usage	Description
CLICK	CLICK $[x, y]$	Click on the coordinate point $[x, y]$.
SCROLL	SCROLL [UP/DOWN/LEFT/RIGHT]	Scroll in the specified direction.
TYPE	TYPE [content]	Type the given content.
OPENAPP	OPENAPP [app_name]	Open an app named [app_name].
COMPLETE	COMPLETE	Mark the current task as completed.
WAIT	WAIT	Wait for the page or content to finish loading.
PRESS_BACK	PRESS_BACK	Press the back button to return to the previous page.
PRESS_HOME	PRESS_HOME	Press the home button to return to the home screen.
PRESS_ENTER	PRESS_ENTER	Press the Enter key.

our experiments is provided in Table 3. An action is considered an exact match only when both t and p exactly match the ground truth. The calculation of AMR varies depending on the action type, as outlined below:

For actions without parameters (e.g., COMPLETE), evaluation depends only on action type t. AMR is equivalent to type match rate (TMR) for these actions.

For SCROLL, we evaluate both action type t and scroll direction (UP, DOWN, LEFT, or RIGHT) to ensure they perfectly align with the ground truth.

For TYPE, we adopt a comparatively less stringent evaluation. After verifying that the predicted action type t is TYPE, both the ground truth and predicted text are converted to lowercase and trimmed of leading and trailing spaces. The action is considered a match if the normalized predicted text exactly matches the normalized ground truth.

For OPENAPP, we also adopt a comparatively less stringent evaluation. This is due to ambiguity in app names (e.g., voice recorder-unrecorder vs. voice recorder) and inconsistencies between ground truth actions and the low-level instructions of AndroidControl (e.g. OPENAPP *Flipsnack* vs. *open the flipsnack magazine app*). Specifically, we first verify that the predicted action type t is OPENAPP, then normalize all words in the predicted and ground truth app names by converting them into lowercase and applying stemming to reduce variations in tense and person. If either normalized app name is a substring of the other, the action is considered an exact match.

For CLICK actions, we slightly modify the evaluation method from OS-Atlas [16], leveraging the availability of widget bounding boxes. Specifically, when both the predicted and ground truth action types t are CLICK, we first inspect the corresponding screenshot layout to identify the bounding box b containing the ground truth coordinates $[x, y]$. If such a box exists, the action is considered correct if the predicted coordinates $[\hat{x}, \hat{y}]$ fall within b; otherwise, we measure the relative distance. If no bounding box is found, correctness is determined solely by the relative distance $d = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2}$ between the predicted and ground truth coordinates. For the state control benchmark, the fine-grained nature of GUI toggles makes the commonly adopted 14% threshold overly permissive, as even such deviations may lead to failed toggling. We therefore consider a toggle action correct only if the relative distance is below 4% of the screen (i.e., $d \leq 40$ in our normalized coordinate system). For agentic benchmarks, we maintain the commonly adopted 14% threshold (i.e., $d \leq 140$), as agentic tasks generally tolerate higher deviation.

A.3. Details of Agentic Benchmark

The agentic benchmarks adopted in this paper are:

• AndroidControl [6] is a large-scale mobile agent benchmark comprising 15,283 demonstrations with step-wise instructions. Data are collected from human raters performing various tasks on 833 apps across 40 categories on Android devices. The training subset of AndroidControl includes 89,144 step-wise samples.

• AITZ [18] is a mobile agent benchmark derived from a subset of AITW [12] and annotated by GPT-4o [10] for chain-of-action-thought (CoAT) components. It includes 2,504 operation trajectories across 18,643 steps, categorized into five domains: General, Install, GoogleApps, Single, and Web Shopping. The training subset of AITZ contains 13,919 step-wise samples.

• GUI-Odyssey [13] is a large-scale mobile benchmark for training and evaluating cross-app navigation agents on complex, long-chain tasks. It consists of 8,334 episodes from 6 mobile devices, covering 6 cross-app task types, 212 apps, and over 1,400 app combinations. The training subset of GUI-Odyssey includes 101,486 step-wise samples.

The complementary characteristics of these benchmarks enable comprehensive evaluation of agent capabilities across multiple dimensions: AndroidControl provides broad coverage and generalization across applications, AITZ provides explicit reasoning traces with detailed annotations, and GUI Odyssey emphasizes complex long

Table 4. Statistical information for the test subsets of all three agentic benchmarks.

Benchmark	CLICK	COMPLETE	SCROLL	TYPE	OPENAPP	PRESS	Others	Total Step	Trajectory
AndroidControl	5074	1543	1211	632	608	343	576	9987	1543
AITZ	2736	504	601	500	/	265	118	4724	506
GUI-Odyssey	16658	1572	2622	2666	/	2044	89	25651	1666

Table 5. Task name and user instruction templates of the dynamic evaluation benchmark.

Task Name	User Instruction Template
SystemBluetoothTurnOff	Turn bluetooth off.
SystemBluetoothTurnOffVerify	Turn bluetooth off.
SystemBluetoothTurnOn	Turn bluetooth on.
SystemBluetoothTurnOnVerify	Turn bluetooth on.
SystemWifiTurnOff	Turn wifi off.
SystemWifiTurnOffVerify	Turn wifi off.
SystemWifiTurnOn	Turn wifi on.
SystemWifiTurnOnVerify	Turn wifi on.
TurnOffWifiAndTurnOnBluetooth	Turn off WiFi, then enable bluetooth
TurnOnWifiAndOpenApp	Turn on Wifi, then open the {app_name} app
TurnOnAlarm9AM	Trun on alarm at 9:00 AM.
TurnOffAlarm9AM	Trun off alarm at 9:00 AM.
TurnOnCaptionYoutube	Turn on captions in Youtube's settings.
TurnOffCaptionYoutube	Turn off captions in Youtube's settings.
TurnOnDoNotDisturb	Turn on Do not Disturb
TurnOffDoNotDisturb	Turn off Do not Disturb
TurnOnSaveAndFillPaymentMethodsChrome	Turn on save and fill payment methods in Chrome's settings.
TurnOffSaveAndFillPaymentMethodsChrome	Turn off save and fill payment methods in Chrome's settings.
TurnOnAlwaysSecureConnChrome	Turn on Always use the secure connections in Chrome's settings.
TurnOffAlwaysSecureConnChrome	Turn off Always use the secure connections in Chrome's settings.

horizon tasks. We present detailed statistics of the test subsets for all three benchmarks in Table 4.

A.4. Details of Dynamic Evaluation Benchmark

To evaluate the real-world applicability of StaR, we construct a dynamic evaluation benchmark consisting of 20 real-world toggle control tasks selected from daily mobile usage scenarios. This benchmark assesses three key aspects: (i) overall task execution accuracy, (ii) false positive toggling when the current toggle state already matches the desired state (Verify cases), and (iii) false negative toggling in normal operation scenarios.

The benchmark is implemented on the Android emulator provided by AndroidStudio and built upon the AndroidWorld [14] framework. We design two types of evaluation scenarios for each toggle operation: normal execution cases and verification cases (marked with *Verify* suffix) where the current toggle state already matches the desired state. Table 5 presents all task names and user instruction templates, covering system settings (e.g., WiFi/Bluetooth), applications (Youtube/Chrome/Alarm), and composite tasks.

Following AndroidWorld [14], we adopt overall task success rate ranging from $[0, 1]$ as the primary metric, calculated as the ratio of successfully completed subtasks. Overall task success rate primary reflects overall task execution accuracy and reveal the severity of false positives and false negatives in this benchmark. For composite tasks like "Turn off WiFi, then enable Bluetooth", the task success rate reflects partial completion (e.g., 0.5 if one subtask fails).

A.5. Implementation Details of StaR

Rather than prompting, we train the multimodal agents to explicitly learn the StaR reasoning process. To improve the abilities on toggle state control tasks, we include the training split of the state control benchmark and adjust the reasoning process of each sample into StaR style. Additionally, to preserve performance on general agentic tasks and to learn to apply StaR reasoning adaptively in toggle state control tasks, we also annotate commonly adopted agentic benchmarks. Specifically, for those episodes representing toggle state control tasks, we identify the concrete step of toggling and refine the corresponding reasoning process into StaR-style. For other steps, we insert the phase "*Target toggle not found in this screen*" into the original reasoning process to help the agents learn to apply StaR reasoning only on critical toggling steps. For episodes that do not

Table 6. Training hyperparameter settings of StaR.

Hyperparameter	Value
finetuning_type	full
freeze_vision_tower	False
freeze_multi_modal_projector	False
cutoff_len	8192
per_device_train_batch_size	1
gradient_accumulation_steps	8
learning_rate	5×10^{-6}
epochs	3
lr_scheduler_type	cosine
warmup_ratio	0.1
parameter_data_type	bf16

represent toggle state control tasks, we directly adopt the original reasoning process. After training on both the state control benchmark and the agentic benchmarks, the multimodal agents learn to apply StaR reasoning adaptively in the critical steps for toggling and to retain the original reasoning in other steps, improving toggle accuracy without sacrificing general agentic performance.

To train multimodal agents, we adopt their respective original prompt templates and click coordinate settings. For OS-Atlas-7B [16], UI-TARS-7B [11], and AgentCPM-GUI-8B [19], the click coordinates are normalized to $[0, 1000]$. For GUI-OWL-7B [17], the click coordinates are set to original pixel coordinates. We adopt the LLaMA-Factory [20] framework to train the multimodal agents, with detailed training hyperparameters provided in Table 6. Additionally, FlashAttention [4] is adopted for acceleration.

For evaluation, as each multimodal agent has its own action format, we translate the action format into OS-Atlas-style [16] for unified evaluation. For UI-TARS, which adopts multi screenshot history modeling, we evaluate each episode independently, adopting the previously predicted four steps within the same episode as the historical input. For general proprietary MLLM-based agents that lack dedicated GUI-agentic prompts, we adopt the UI-TARS prompt due to its simplicity. Similarly, we adopt the OS-Atlas prompt [16] for Qwen-2-VL-72B [15]. For GUI-R1 [9], which is built on Qwen-2.5-VL-7B [1], we follow its original prompt configuration and set the click coordinates to original pixel coordinates.

B. Additional Results and Analyses

This section reports additional experimental results and analyses. Section B.1 provides detailed evaluations of existing multimodal agents on the state control benchmark with more comprehensive comparisons to four training free baselines, highlighting the effectiveness of StaR and the necessity of training. Section B.2 reports results and analysis for omitted agents on agentic benchmarks, demonstrating that StaR training improves or maintains performance on general agentic tasks. Section B.3 presents ablation studies on StaR components, indicating that StaR is most effective when all three components are integrated. Section B.4 provides case studies illustrating how StaR improves perception, reasoning, and execution in toggle control tasks.

B.1. Detailed Evaluation Results of Existing Multimodal Agents on State Control Benchmark

We present the detailed evaluation results of existing multimodal agents on the state control benchmark under different settings in Table 7. Specifically, in addition to the vanilla zero-shot baseline, we include four training-free baselines:
(i) **State-focused Prompt Engineering,** where we append additional prompt to the original prompt to guide the agents to focus on toggle state during reasoning (see Section 3.2), with the prompt template in Appendix C.
(ii) **StaR-style Prompting**, where we append additional prompt to the original prompt to guide the agents to follow the StaR reasoning process (see Section 5.2), with the prompt template in Appendix C.
(iii) **Ground Truth Toggle State Prompting**, where we provide the ground truth current toggle state in the prompt, representing the **theoretical upper bound** of performance that a multi-agent collaboration framework achieves when an additional annotator agent identifies the toggle state with perfect accuracy and guides the reasoning of the action agent. The prompt template is provided in Appendix C.
(iv) **Ground Truth Toggle State and StaR-style Prompting**, where we provide the ground truth current toggle state in the prompt and append additional prompt to the original prompt to guide the agents to follow the StaR reasoning process. This baseline reflects the **theoretical upper bound** of performance achievable by a multi-agent collaboration framework that introduces an additional annotator agent to identify the toggle state with perfect accuracy and guide the reasoning of the action agent, along with following the StaR reasoning process. In other words, this baseline can reflect the **theoretical upper bound without training**.

Based on the results, we draw the following conclusions:

(i) Existing multimodal agents perform poorly in state control tasks. Even the best-performing agent yield less than 70% O-AMR, and most agents yield less than 50% O-AMR, which is close to random toggling. While these agents perform better on positive samples, their grounding ability remains limited, with P-AMR generally below 60%. For negative samples, all agents show a strong bias toward false positive toggling, resulting in high N-FPTR. These results demonstrate the limited capability of current multimodal agents in state control.

(ii) All baselines without training do not fundamen-

Table 7. Detailed evaluation results of existing multimodal agents on state control benchmark. Subscripts denote absolute improvements over the zero-shot baseline, with red indicating improvements and green indicating degradations. The optimal and the suboptimal results are **bolded** and underlined, respectively. Results demonstrate that StaR training significantly improves execution and grounding accuracy on state control benchmark, outperforming prompting baselines significantly and highlighting the necessity of training.

Model	O-TMR↑	O-AMR↑	P-TMR↑	P-AMR↑	P-FNR↓	N-AMR↑	N-FPTR↓	N-FPR↓
Zero-shot								
GPT-5	75.35	37.05	91.91	15.30	2.79	58.80	36.14	3.01
GPT-4o	72.40	27.17	97.04	6.60	2.35	47.75	48.83	**2.39**
Gemini-2.5-Pro	68.74	30.25	**98.85**	21.87	**0.78**	38.64	60.14	9.31
Qwen-2-VL-72B	**87.59**	**66.42**	96.21	53.89	3.69	**78.96**	**20.67**	6.33
GUI-R1-7B	78.27	54.14	97.58	49.32	2.03	58.97	40.37	12.63
OS-Atlas-7B	67.16	43.95	98.51	52.10	1.27	35.80	64.10	28.67
UI-TARS-7B	67.14	47.45	94.33	54.94	1.71	39.96	48.29	17.62
AgentCPM-GUI-8B	81.74	64.08	95.38	**60.04**	3.32	68.11	30.69	11.07
GUI-Owl-7B	76.58	53.57	94.99	48.97	2.32	58.16	39.15	14.66
w/ State-focused Prompt Engineering								
GPT-5	82.09$_{\uparrow 6.74}$	46.35$_{\uparrow 9.30}$	87.32$_{\downarrow 4.59}$	15.84$_{\uparrow 0.54}$	4.25$_{\uparrow 1.46}$	76.86$_{\uparrow 18.06}$	**15.00**$_{\downarrow 21.14}$	1.22$_{\downarrow 1.79}$
GPT-4o	82.87$_{\uparrow 10.47}$	38.66$_{\uparrow 11.49}$	94.01$_{\downarrow 3.03}$	5.60$_{\downarrow 1.00}$	4.79$_{\uparrow 2.44}$	71.73$_{\uparrow 23.98}$	25.88$_{\downarrow 22.95}$	**1.03**$_{\downarrow 1.36}$
Gemini-2.5-Pro	81.78$_{\uparrow 13.04}$	42.86$_{\uparrow 12.61}$	**97.29**$_{\downarrow 1.56}$	19.45$_{\downarrow 2.42}$	**2.39**$_{\uparrow 1.61}$	66.28$_{\uparrow 27.64}$	33.06$_{\downarrow 27.08}$	3.74$_{\downarrow 5.57}$
Qwen-2-VL-72B	87.11$_{\downarrow 0.48}$	65.29$_{\downarrow 0.36}$	95.85$_{\downarrow 1.69}$	52.20$_{\uparrow 0.34}$	4.03$_{\uparrow 0.34}$	78.37$_{\downarrow 0.59}$	20.99$_{\uparrow 0.32}$	7.87$_{\uparrow 1.54}$
GUI-R1-7B	**89.15**$_{\uparrow 10.88}$	**65.59**$_{\uparrow 11.45}$	94.47$_{\downarrow 3.11}$	47.65$_{\downarrow 1.67}$	4.99$_{\uparrow 2.96}$	**83.53**$_{\uparrow 24.56}$	16.13$_{\downarrow 24.24}$	4.37$_{\downarrow 8.26}$
OS-Atlas-7B	75.21$_{\uparrow 8.05}$	52.55$_{\uparrow 8.60}$	94.53$_{\downarrow 3.98}$	49.22$_{\downarrow 2.88}$	4.42$_{\uparrow 3.15}$	55.89$_{\uparrow 20.09}$	43.57$_{\downarrow 20.53}$	18.91$_{\downarrow 9.76}$
UI-TARS-7B	81.84$_{\uparrow 14.70}$	63.23$_{\uparrow 15.78}$	93.38$_{\downarrow 0.95}$	56.16$_{\uparrow 1.22}$	5.91$_{\uparrow 4.20}$	70.31$_{\uparrow 30.35}$	27.83$_{\downarrow 20.46}$	9.38$_{\downarrow 8.24}$
AgentCPM-GUI-8B	82.30$_{\uparrow 0.63}$	64.67$_{\uparrow 0.59}$	95.43$_{\uparrow 0.05}$	**60.04**$_{\downarrow 0.00}$	3.47$_{\uparrow 0.15}$	69.31$_{\uparrow 1.20}$	30.01$_{\downarrow 0.68}$	10.65$_{\downarrow 0.42}$
GUI-Owl-7B	85.02$_{\uparrow 8.44}$	61.09$_{\uparrow 7.52}$	96.02$_{\uparrow 1.03}$	48.17$_{\downarrow 0.80}$	3.05$_{\uparrow 0.73}$	74.02$_{\uparrow 15.86}$	24.51$_{\downarrow 14.64}$	6.96$_{\downarrow 7.70}$
w/ StaR-style Prompting								
Qwen-2-VL-72B	**87.81**$_{\uparrow 0.22}$	**65.91**$_{\downarrow 0.51}$	95.50$_{\downarrow 0.71}$	51.71$_{\downarrow 2.18}$	4.37$_{\uparrow 0.68}$	80.11$_{\uparrow 1.15}$	19.67$_{\downarrow 1.00}$	6.33$_{\uparrow 0.00}$
GUI-R1	87.77$_{\uparrow 9.50}$	65.10$_{\uparrow 10.96}$	95.58$_{\downarrow 2.00}$	48.24$_{\downarrow 1.08}$	4.18$_{\uparrow 2.15}$	**81.96**$_{\uparrow 22.99}$	**17.77**$_{\downarrow 22.60}$	**5.47**$_{\downarrow 7.16}$
OS-Atlas-7B	73.52$_{\uparrow 6.36}$	50.07$_{\uparrow 6.12}$	**96.77**$_{\downarrow 1.74}$	49.88$_{\downarrow 2.22}$	**2.96**$_{\uparrow 1.69}$	50.27$_{\uparrow 14.47}$	49.62$_{\downarrow 14.48}$	22.21$_{\downarrow 6.46}$
UI-TARS-7B	81.18$_{\uparrow 14.04}$	62.89$_{\uparrow 15.44}$	90.98$_{\uparrow 3.35}$	54.40$_{\uparrow 0.54}$	8.55$_{\uparrow 6.84}$	71.38$_{\uparrow 31.42}$	27.54$_{\downarrow 20.75}$	9.38$_{\downarrow 8.24}$
AgentCPM-GUI-8B	82.14$_{\uparrow 0.40}$	64.43$_{\uparrow 0.35}$	95.36$_{\downarrow 0.02}$	**59.95**$_{\downarrow 0.09}$	3.59$_{\uparrow 0.27}$	68.91$_{\uparrow 0.80}$	30.28$_{\downarrow 0.41}$	10.58$_{\downarrow 0.49}$
GUI-Owl-7B	84.27$_{\uparrow 7.69}$	60.92$_{\uparrow 7.35}$	94.06$_{\downarrow 0.93}$	47.36$_{\downarrow 1.61}$	4.55$_{\uparrow 2.23}$	74.49$_{\uparrow 16.33}$	23.68$_{\downarrow 15.47}$	7.48$_{\downarrow 7.18}$
w/ Ground Truth Toggle State Prompting								
Qwen-2-VL-72B	**94.00**$_{\uparrow 6.41}$	**72.14**$_{\uparrow 5.72}$	95.80$_{\downarrow 0.41}$	52.08$_{\downarrow 1.81}$	4.13$_{\uparrow 0.44}$	**92.20**$_{\uparrow 13.24}$	**7.70**$_{\downarrow 12.97}$	**2.37**$_{\downarrow 3.96}$
GUI-R1	85.26$_{\uparrow 6.99}$	60.63$_{\uparrow 6.49}$	97.63$_{\uparrow 0.05}$	48.36$_{\downarrow 0.96}$	1.98$_{\downarrow 0.05}$	72.90$_{\uparrow 13.93}$	26.78$_{\downarrow 13.59}$	7.87$_{\downarrow 4.76}$
OS-Atlas-7B	68.29$_{\uparrow 1.13}$	45.22$_{\uparrow 1.27}$	**98.46**$_{\downarrow 0.05}$	52.32$_{\uparrow 0.22}$	**1.34**$_{\uparrow 0.07}$	38.12$_{\uparrow 2.32}$	61.73$_{\downarrow 2.37}$	28.15$_{\downarrow 0.52}$
UI-TARS-7B	73.53$_{\uparrow 6.39}$	54.03$_{\uparrow 6.58}$	96.11$_{\uparrow 1.78}$	57.18$_{\uparrow 2.24}$	1.81$_{\uparrow 0.10}$	50.88$_{\uparrow 10.92}$	42.67$_{\downarrow 5.62}$	15.18$_{\downarrow 2.44}$
AgentCPM-GUI-8B	82.48$_{\uparrow 0.74}$	64.88$_{\uparrow 0.80}$	95.58$_{\uparrow 0.20}$	**60.39**$_{\uparrow 0.35}$	3.27$_{\downarrow 0.05}$	69.38$_{\uparrow 1.27}$	29.99$_{\downarrow 0.70}$	10.63$_{\downarrow 0.44}$
GUI-Owl-7B	83.22$_{\uparrow 6.64}$	60.07$_{\uparrow 6.50}$	96.68$_{\uparrow 1.69}$	50.37$_{\uparrow 1.40}$	1.91$_{\downarrow 0.41}$	69.77$_{\uparrow 11.61}$	28.18$_{\downarrow 10.97}$	9.19$_{\downarrow 5.47}$
w/ Ground Truth Toggle State and StaR-style Prompting								
Qwen-2-VL-72B	**92.85**$_{\uparrow 5.26}$	**70.88**$_{\uparrow 4.46}$	96.21$_{\downarrow 0.00}$	52.27$_{\downarrow 1.62}$	3.59$_{\downarrow 0.10}$	**89.49**$_{\uparrow 10.53}$	**10.26**$_{\downarrow 10.41}$	**2.86**$_{\downarrow 3.47}$
GUI-R1	90.24$_{\uparrow 11.97}$	66.31$_{\uparrow 12.17}$	95.38$_{\downarrow 2.20}$	47.53$_{\downarrow 1.79}$	4.45$_{\uparrow 2.42}$	85.09$_{\uparrow 26.12}$	14.81$_{\downarrow 25.56}$	4.57$_{\downarrow 8.06}$
OS-Atlas-7B	75.55$_{\uparrow 8.39}$	51.78$_{\uparrow 7.83}$	97.78$_{\downarrow 0.73}$	50.24$_{\downarrow 1.86}$	2.15$_{\uparrow 0.88}$	53.32$_{\uparrow 17.52}$	46.60$_{\downarrow 17.50}$	19.97$_{\downarrow 8.70}$
UI-TARS-7B	84.04$_{\uparrow 16.90}$	65.29$_{\uparrow 17.84}$	95.41$_{\uparrow 1.08}$	57.89$_{\uparrow 2.95}$	4.08$_{\uparrow 2.37}$	72.68$_{\uparrow 32.72}$	26.56$_{\downarrow 21.73}$	8.75$_{\downarrow 8.87}$
AgentCPM-GUI-8B	82.20$_{\uparrow 0.46}$	64.37$_{\uparrow 0.29}$	95.50$_{\uparrow 0.12}$	**59.85**$_{\downarrow 0.19}$	3.35$_{\uparrow 0.03}$	68.89$_{\uparrow 0.78}$	30.40$_{\downarrow 0.29}$	10.78$_{\downarrow 0.29}$
GUI-Owl-7B	78.29$_{\uparrow 1.71}$	54.70$_{\uparrow 1.13}$	**98.00**$_{\uparrow 3.01}$	50.83$_{\uparrow 1.86}$	**1.44**$_{\downarrow 0.88}$	58.58$_{\uparrow 0.42}$	40.57$_{\uparrow 1.42}$	15.66$_{\uparrow 1.00}$
w/ StaR Training								
OS-Atlas-7B	**96.13**$_{\uparrow 28.97}$	**79.72**$_{\uparrow 35.77}$	**95.77**$_{\downarrow 2.74}$	**62.95**$_{\uparrow 10.85}$	**4.23**$_{\uparrow 2.96}$	96.48$_{\uparrow 60.68}$	3.52$_{\downarrow 60.58}$	1.52$_{\downarrow 27.15}$
UI-TARS-7B	95.82$_{\uparrow 28.68}$	77.86$_{\uparrow 30.41}$	95.11$_{\uparrow 0.78}$	59.19$_{\uparrow 4.25}$	4.89$_{\uparrow 3.18}$	96.53$_{\uparrow 56.57}$	3.45$_{\downarrow 44.84}$	1.34$_{\downarrow 16.28}$
AgentCPM-GUI-8B	95.98$_{\uparrow 14.24}$	79.00$_{\uparrow 14.92}$	94.50$_{\downarrow 0.88}$	60.53$_{\uparrow 0.49}$	5.50$_{\uparrow 2.18}$	**97.46**$_{\uparrow 29.35}$	**2.54**$_{\downarrow 28.15}$	**0.95**$_{\downarrow 10.12}$
GUI-Owl-7B	95.99$_{\uparrow 19.41}$	77.60$_{\uparrow 24.03}$	95.65$_{\uparrow 0.66}$	58.87$_{\uparrow 9.90}$	**4.35**$_{\uparrow 2.03}$	96.33$_{\uparrow 38.17}$	3.67$_{\downarrow 35.48}$	1.56$_{\downarrow 13.10}$

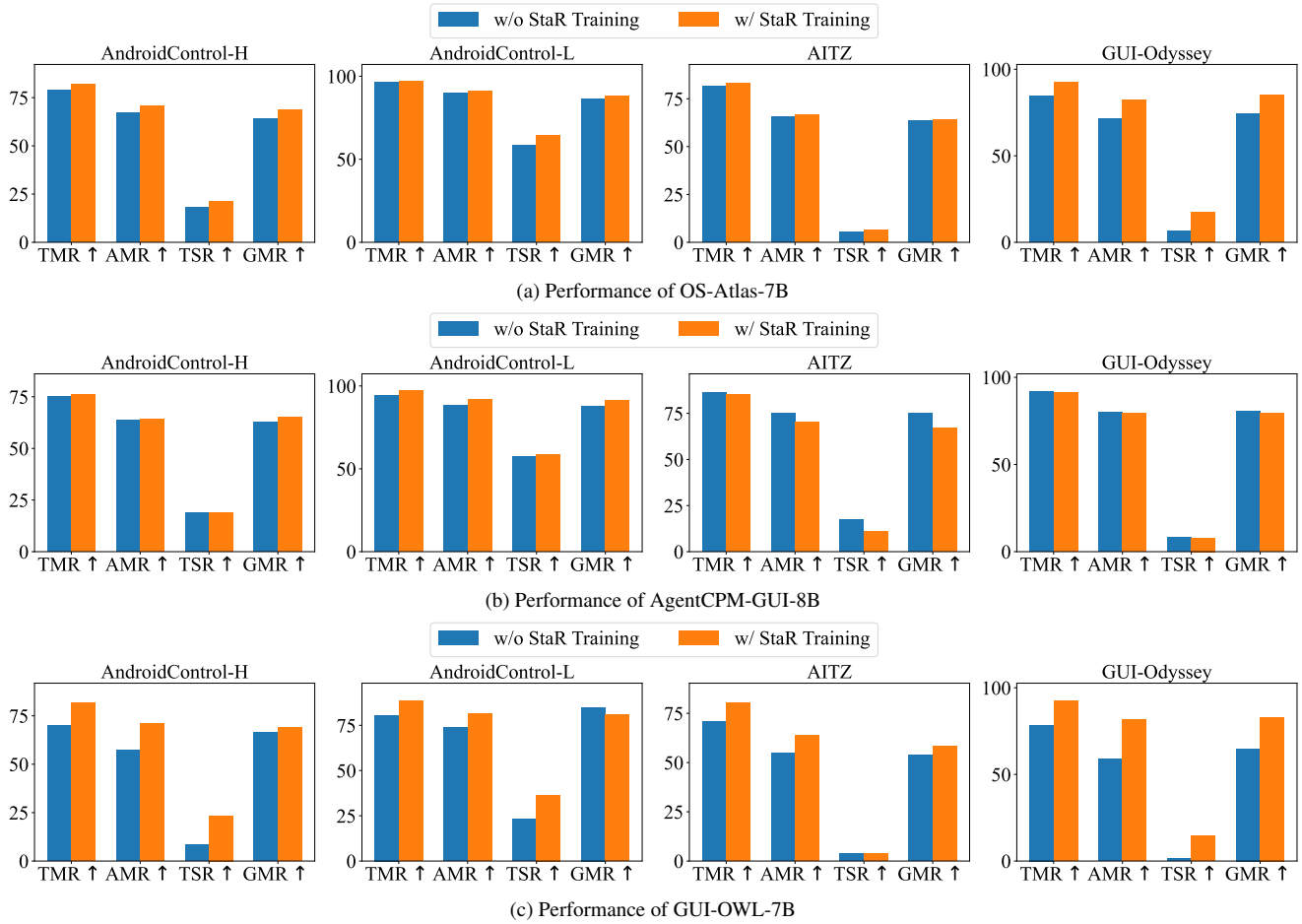

Figure 2. The performance o f zero-shot and StaR-trained (a) OS-Atlas-7B; (b) AgentCPM-GUI-8B; and (c) GUI-OWL-7B on agentic benchmarks. Results demonstrate that StaR consistently preserves or enhances performance on agentic benchmarks.

tally improve performance. For all baselines, weak agents such as OS-Atlas-7B and UI-TARS-7B achieves notable improvements on negative samples, but their overall performance remains unsatisfactory. For strong agents such as Qwen-2-VL-72B and AgentCPM-GUI-8B, the improvements are marginal and may even result in degradations. This is likely attributed to two reasons. On one hand, prompting does not fundamentally improve the grounding ability of multimodal agents, and the recognition of GUI toggles remains poor. On the other hand, prompting is less effective in establishing the mapping between negative instructions and the corresponding correct actions, resulting in improved but still unsatisfactory performance on negative samples. Collectively, These results suggest that relying solely on prompting or multi-agent annotation collaboration without training is not sufficient to address the limitations of multimodal agents in state control.

(iii) Only StaR training leads to substantial improvements, highlighting the necessity of training. StaR training consistently improves both O-AMR and P-AMR, surpassing that of all baselines significantly. For positive samples, StaR training can yield improvements on P-AMR, likely due to training on toggle clicks that strengthen toggle recognition. In contrast, P-AMR generally decreases under prompting-based baselines. For negative samples, StaR training achieves nearly 100% N-AMR, significantly mitigating false positive toggling and surpassing all baselines. These findings highlight the effectiveness of StaR training in improving performance on both positive and negative samples, reinforcing the necessity of training to learn this state-aware structured reasoning process.

Collectively, StaR training demonstrates its effectiveness in improving the execution and grounding capabilities of multimodal agents on the state control benchmark.

B.2. Detailed Performance on Agentic Benchmarks

We present the detailed performance of the omitted agents on the agentic benchmarks in Figure 2.

Table 8. Ablation results of OS-Atlas-7B with different StaR training components, where P, A, D represent perceiving, analyzing, and deciding, respectively. Subscripts denote absolute improvements over the vanilla baseline, with red indicating improvements and green indicating degradations. The optimal results are **bolded**. Removing perceiving or analyzing consistently degrades performance, confirming that StaR is most effective when all three components are integrated.

Method	Components P	A	D	O-TMR↑	O-AMR↑	P-TMR↑	P-AMR↑	P-FNR↓	N-AMR↑	N-FPTR↓	N-FPR↓
Vanilla	×	×	×	67.16	43.95	98.51	52.10	1.27	35.80	64.10	28.67
StaR w/o Perceiving	×	✓	✓	$89.97_{\uparrow 22.81}$	$73.39_{\uparrow 29.44}$	$95.94_{\downarrow 2.57}$	$62.78_{\uparrow 10.68}$	**4.06**$_{\uparrow 2.79}$	$83.99_{\uparrow 48.19}$	$16.01_{\downarrow 48.09}$	$9.41_{\downarrow 19.26}$
StaR w/o Analyzing	✓	×	✓	$90.57_{\uparrow 23.41}$	$73.94_{\uparrow 29.99}$	$95.63_{\downarrow 2.88}$	$62.32_{\uparrow 10.22}$	$4.35_{\uparrow 3.08}$	$85.56_{\uparrow 49.76}$	$14.42_{\downarrow 49.68}$	$9.92_{\downarrow 18.75}$
StaR	✓	✓	✓	**96.13**$_{\uparrow 28.97}$	**79.72**$_{\uparrow 35.77}$	**95.77**$_{\downarrow 2.74}$	**62.95**$_{\uparrow 10.85}$	$4.23_{\uparrow 2.96}$	**96.48**$_{\uparrow 60.68}$	**3.52**$_{\downarrow 60.58}$	**1.52**$_{\downarrow 27.15}$

From the results, we have the following findings:

(i) StaR consistently improves or preserves general agentic performance. Across four benchmark settings, StaR training consistently preserves or surpasses baselines. While AgentCPM-GUI-8B shows outlier degradations on AITZ and GUI-Odyssey, our error analysis on 463 degradation and 287 improvement cases before and after StaR training reveals that 28% of degradations are due to coordinate drift and 37% to valid alternative operations (e.g., clicking the back icon vs. PRESS_BACK). As alternatives appear in both cases, coordinate drift is the primary cause. The drift likely stems from the fixed visual tokens of AgentCPM, which are less robust than the flexible tokens of Qwen-2-VL to the large resolution span of AITZ and GUI-Odyssey. Thus, this outlier reflects model-specific grounding sensitivity distinct from StaR effectiveness.

(ii) StaR generalizes across multimodal agents. Results on all agents indicate that StaR consistently improves performance on agentic benchmarks. These findings demonstrate that StaR is model-agnostic and can effectively enhance the reasoning ability of diverse multimodal agents.

Overall, the results indicate that StaR training does not compromise model capability. Additionally, in several settings, StaR leads to measurable improvements, highlighting its generalizability on general agentic tasks.

B.3. Ablation Studies on StaR components

To further evaluate the impact of each component (perceiving, analyzing, deciding) in the StaR reasoning chain for toggle control, we select OS-Atlas-7B as the target agent and train it with different combinations of these components. Since decision is generally equivalent to low-level instructions, we only consider removing perceiving or analyzing. The ablation results are presented in Table 8.

From the results, we observe the following:

(i) Removing perceiving or analyzing consistently degrades performance. Additionally, Ground Truth Toggle State Prompting in Appendix B.1 can be approximately regarded as making action decisions based solely on perception. The corresponding results are still inferior to those of StaR training, which integrates all three reasoning components. These findings confirm that StaR is most effective when all three components are integrated.

(ii) StaR training, even with some components removed, still outperforms the vanilla baseline, demonstrating its effectiveness in enabling precise toggle control.

B.4. Case Studies

To demonstrate the effectiveness of StaR-trained agents in precisely executing real-world toggle control instructions, we adopt OS-Atlas-7B, which exhibited the most pronounced improvement in Section 5.4, as a representative example. The instruction is "*turn wifi on*", with the toggle initially set to *on*, thereby testing false-positive toggling. The trajectories of OS-Atlas-7B without and with StaR training are presented in Figure 3 and Figure 4, respectively.

From these examples, we observe the following:

(i) Without StaR training, OS-Atlas-7B fails to execute the instruction correctly, resulting in a false positive toggle. The agent misperceives the current toggle state as *off* and incorrectly clicks the toggle, resulting in an unintended state change. It then repeatedly toggles between *on* and *off*, falling into an infinite loop and ultimately failing the task.

(ii) With StaR training, OS-Atlas-7B successfully executes the instruction correctly. At the critical decision step, the agent adaptively applies the state-aware reasoning chain, correctly perceiving the current toggle state as *on* and appropriately deciding to finish the task, thereby completing the instruction as intended.

These case studies illustrate the effectiveness of StaR in enabling agents to precisely execute real-world toggle control instructions in dynamic environments.

C. Prompts

This section presents the meticulously designed prompt templates. The template for toggle identification, state-functionality annotation, UI-TARS, OS-Atlas, AgentCPM-GUI, and GUI-Owl are provided in Figure 5, Figure 6, Figure 7, Figure 8, Figure 9, and Figure 10, respectively.

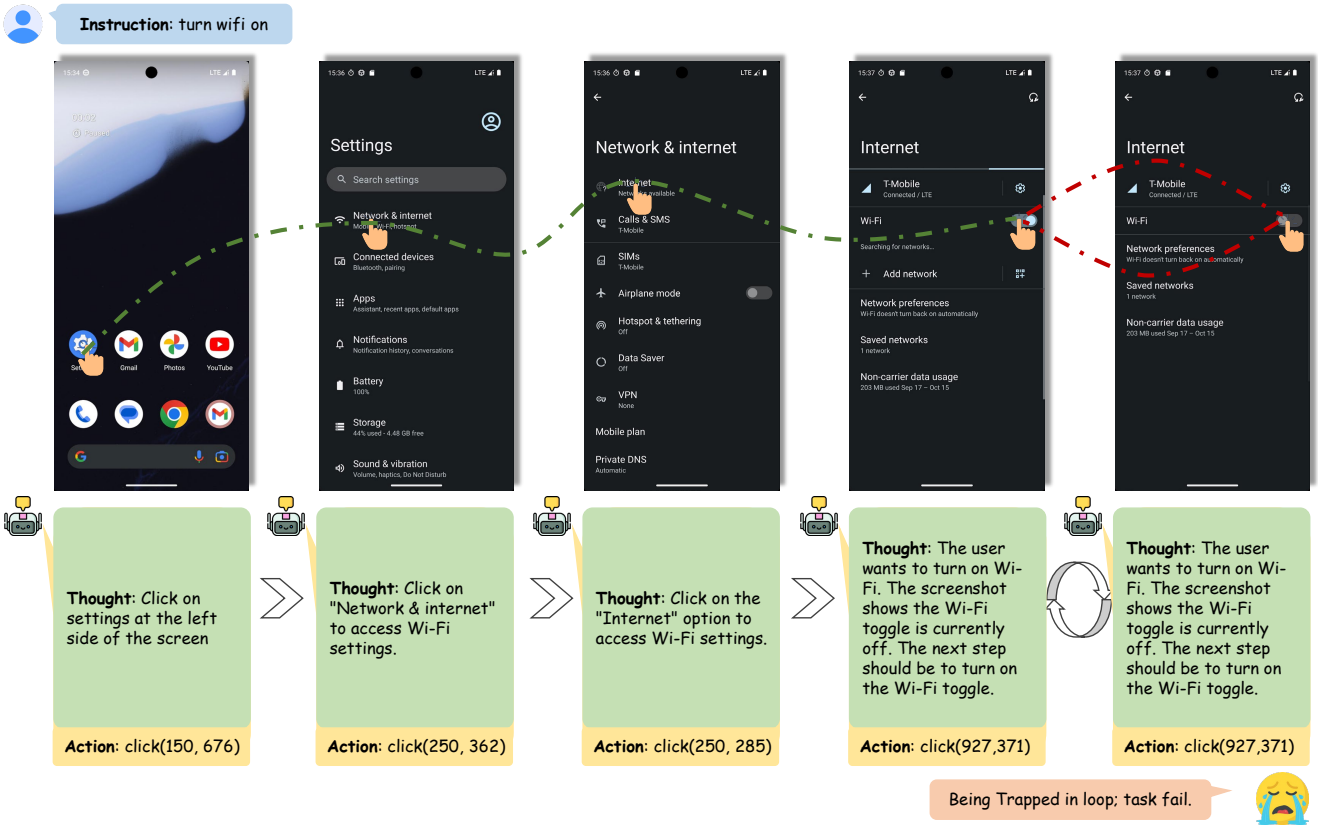

Figure 3. Trajectory of OS-Atlas-7B without StaR training when executing the instruction *turn wifi on* with the toggle initially *on*. The agent mistakenly perceives the current toggle state as *off* and incorrectly clicks the toggle, resulting in an unintended state change. It then repeatedly toggles between *on* and *off*, falling into an infinite loop and ultimately failing the task.

For the two prompting baselines in Section 3.2 and Section 5.2, we append supplementary templates to the original prompts to guide the reasoning. The prompt templates for State-focused Prompt Engineering, and StaR-style Prompting are provided in Figure 11 and Figure 12, respectively. For ground truth toggle state prompting in Appendix B.1, we also append supplementary templates to the original prompts to provide current toggle state and guide the reasoning. The prompt templates for ground truth toggle state prompting are provided in Figure 13.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 6

[2] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2138–2156, Vienna, Austria, 2025. 1, 2

[3] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language model to versatile gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21936–21959, Vienna, Austria, 2025. 1, 2

[4] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 6

[5] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024. 1

[6] Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. 4

[7] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. In *Proceedings of the 2020 Conference on Empirical Methods in*

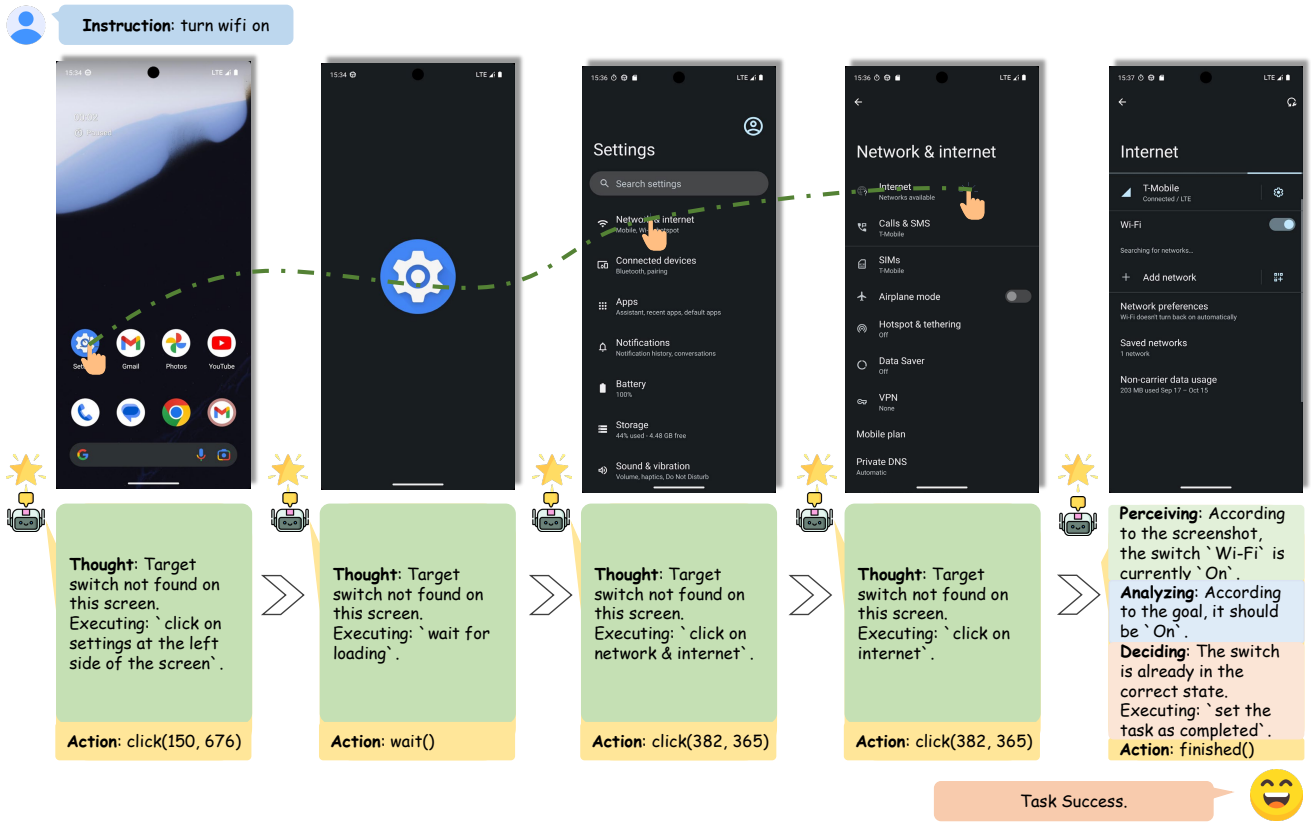

Figure 4. Trajectory of OS-Atlas-7B with StaR training when executing the instruction *turn wifi on* with the toggle initially *on*. At the critical decision step, the agent adaptively applies the state-aware reasoning chain, correctly perceiving the current toggle state as *on* and appropriately deciding to finish the task, thereby completing the instruction as intended.

Natural Language Processing (EMNLP), pages 5495–5510, Online, 2020. Association for Computational Linguistics. 1, 2

[8] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*, 2024. 1

[9] Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025. 6

[10] OpenAI. Gpt-4 system card. 2024. 4

[11] 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. 6

[12] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: a large-scale dataset for android device control. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 59708–59728, New Orleans, LA, USA, 2023. 1, 2, 4

[13] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, Singapore, 2025. 1, 2, 4

[14] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, Singapore, 2025. 5

[15] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 1, 6

[16] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: Foundation action model for generalist GUI agents. In *The Thirteenth International Conference on Learning Representations*, Sin-

Prompt Template for Toggle Identification

```
You are in Switch Detection Mode. Your task is to determine whether the UI element fully enclosed in the red box (__BBOX__) is a switch-type control.

# Inputs:
1. Screenshot: <image> (red box highlights the target)
2. Click Action: "CLICK within red box __BBOX__"
3. Instruction: __INSTRUCTION__ (only extract the object being acted on; ignore action verbs like "enable/disable" and "turn on/off")

# Scope:
- Only analyze the element fully inside the red box.
- Completely ignore surrounding or adjacent elements.
- Do not infer function from layout, instruction, or context.

# Valid Switch Criteria:
An element is a **switch** only if it satisfies **all** of the following:

1. **It is a UI control**, not a label or plain text.
2. **It has a visible binary state** (e.g., ON vs OFF, Enabled vs Disabled, Checked vs Unchecked).
3. **It supports persistent toggle** — clicking must flip state, not trigger one-time action.
4. **It gives visual feedback** that indicates current state before and after the click (e.g., toggle position, color, icon, label change).


# Common Switch Types:
- Checkbox (✓ / empty box)
- Toggle slider
- Dual-state labeled button (e.g., text/icon flips between Enabled / Disabled)

# Not a Switch If:
- The red box contains **plain text**, such as "Vibrate", "Always" or "Choose text color"
- It is a **button** used for **single-use action** (e.g., "Subscribe", "Skip", "Continue")
- It lacks clear binary state indication
- It has no visual change after click
- It is a **static label**, description, or non-interactive UI
- No valid UI element is fully inside the red box

# Rules:
- Do NOT guess based on text like "Subscribe" or "Start"
- Do NOT treat standalone text, button labels, or descriptions as switches
- Do NOT infer behavior from nearby UI elements
- Do NOT treat labels or descriptions as switches
- Only consider visual evidence of **binary toggle capability** inside the red box


# Output Format:
If it is a switch (visual toggle or toggle-button):
{
  "is_switch": true
}
If not:
{
  "is_switch": false
}

Output:
```

Figure 5. Prompt template for toggle identification.

gapore, 2025. 1, 2, 4, 6

[17] Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Foundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025. 6

[18] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. In *Findings of the Association for Computational Linguistics: EMNLP*

Prompt Template for State-functionality Annotation

```
You are in Switch Status Perception Mode. Your task is to determine whether the UI element fully enclosed in the red box (__BBOX__) is a binary switch control that changes state (Enabled or Disabled) when clicked.
# Inputs:
1. Screenshot: <image> (the red box highlights the clickable element)
2. Click Action: "CLICK within red box __BBOX__" (coordinates normalized to [0, 1000])
3. Instruction: __INSTRUCTION__ (used only to name the feature; ignore action verbs like "enable/disable" and "turn on/off")
# Evaluation Scope:
- Analyze ONLY the element fully contained within the red box.
- Completely IGNORE any content outside or partially inside the box.
- Use the instruction ONLY to identify the target feature (noun only, e.g., "Wi-Fi", "Notifications").
- Do NOT infer behavior from instruction verbs or surrounding layout.

# Valid Switch Criteria:
An element is a **switch** only if it satisfies **all** of the following:

1. **It is an interactive UI control**, not a label, icon, or plain text.
2. **It has a visible binary state** (e.g., ON vs OFF, Enabled vs Disabled, Checked vs Unchecked).
3. **It supports persistent toggle** — clicking must flip state reliably, not just trigger an one-time action.
4. **It gives immediate visual feedback** that indicates current state before and after the click (e.g., toggle position, color, icon, label change).

# Common Switch Types:
- Checkbox (✓ / empty box)
- Toggle slider
- Dual-state labeled button (e.g., text/icon flips between Enabled / Disabled)

# Not a Switch If:
- The red box contains **plain text**, such as "Vibrate", "Always" or "Choose text color"
- It is a **button** used for **single-use action** (e.g., "Subscribe", "Skip", "Continue")
- It lacks clear binary state indication
- It has no visual change after click
- It is a **static label**, description, or non-interactive UI
- No valid UI element is fully inside the red box


# Decision Procedure:
1. Locate the UI element fully within the red box.
2. Decide whether it is a switch with binary, persistent states.
3. Extract the controlled feature name from __INSTRUCTION__ (use noun only).
4. Determine the current state **before** the click.
5. Determine the expected state **after** the click.
6. Output structured result as specified.

# Output Format:

If it is a switch and clicking it changes state:
{
  "is_switch": true,
  "feature": "[Feature name in noun form]",
  "state_before_action": "Enabled" or "Disabled",
  "state_after_action": "Enabled" or "Disabled",
  "action_effect": "The action [turn on|turn off] [feature] by changing the switch from [state_before_action] to [state_after_action]"
}

If it is not a switch or does not toggle state:
{
  "is_switch": false,
  "action_effect": "The red box element is not a switch; it triggered [e.g., navigation, selection, or no state change]"
}

# Examples

Instruction: Turn on Bluetooth
{
  "is_switch": true,
  "feature": "Bluetooth",
  "state_before_action": "Disabled",
  "state_after_action": "Enabled",
  "action_effect": "The action turn on Bluetooth by changing the switch from Disabled to Enabled"
}

Instruction: Tap profile icon
{
  "is_switch": false,
  "action_effect": "The red box element is not a switch; it triggered navigation"
}

# Important:
- The red box defines the evaluation boundary — never go outside.
- Do NOT infer meaning from verbs in the instruction.
- Only report a switch if the element shows two visual states and persistent toggle behavior.
- Precision is critical: misclassifying labels or buttons as switches is unacceptable.

Outputs:
```

Figure 6. Prompt template for state-functionality annotation.

2024, pages 12016–12031, Miami, Florida, USA, 2024. 4

[19] Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, Jie Xie, Wei Zhou, Wang Xu, Yuanheng Zhang, Zhou Su, Zhongwu Zhai, Xiaoming Liu, Yudong Mei, Jianming Xu, Hongyan Tian, Chongyi Wang, Chi Chen, Yuan Yao, Zhiyuan Liu, and Maosong Sun. AgentCPM-GUI: Building mobile-use agents with reinforcement fine-tuning. *arXiv preprint arXiv:2506.01391*, 2025. 6

[20] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. 6

Prompt Template for UI-TARS

You are a GUI agent. You are given a task and your action history, with screenshots. You need to perform the next action to complete the task.
Output Format
```
Thought: ...
Action: ...
```
Action Space

click(start_box=\'<|box_start|>(x1,y1)<|box_end|>\')
long_press(start_box=\'<|box_start|>(x1,y1)<|box_end|>\', time=\'\')
type(content=\'\')
scroll(direction=\'down or up or right or left\')
open_app(app_name=\'\')
press_back()
press_home()
wait()
finished() # Submit the task regardless of whether it succeeds or fails.

Note
- Use English in `Thought` part.
- Summarize your next action (with its target element) in one sentence in `Thought` part.

User Instruction
{instruction}

Figure 7. Prompt template for UI-TARS.

Prompt Template for OS-Atlas

You are now operating in Executable Language Grounding mode. Your goal is to help users accomplish tasks by suggesting executable actions that best fit their needs. Your skill set includes both basic and custom actions:
1. Basic Actions
Basic actions are standardized and available across all platforms. They provide essential functionality and are defined with a specific format, ensuring consistency and reliability.
Basic Action 1: CLICK
- purpose: Click at the specified position.
- format: CLICK <point>[[x-axis, y-axis]]</point>
- example usage: CLICK <point>[[101, 872]]</point>

Basic Action 2: TYPE
- purpose: Enter specified text at the designated location.
- format: TYPE [input text]
- example usage: TYPE [Shanghai shopping mall]

Basic Action 3: SCROLL
- purpose: SCROLL in the specified direction.
- format: SCROLL [direction (UP/DOWN/LEFT/RIGHT)]
- example usage: SCROLL [UP]

2. Custom Actions
Custom actions are unique to each user's platform and environment. They allow for flexibility and adaptability, enabling the model to support new and unseen actions defined by users. These actions extend the functionality of the basic set, making the model more versatile and capable of handling specific tasks.

Custom Action 1: PRESS_BACK
- purpose: Press a back button to navigate to the previous screen.
- format: PRESS_BACK
- example usage: PRESS_BACK

Custom Action 2: PRESS_HOME
- purpose: Press a home button to navigate to the home page.
- format: PRESS_HOME
- example usage: PRESS_HOME

Custom Action 3: IMPOSSIBLE
- purpose: Indicate the task is impossible.
- format: IMPOSSIBLE
- example usage: IMPOSSIBLE

Custom Action 4: COMPLETE
- purpose: Indicate the task is finished.
- format: COMPLETE
- example usage: COMPLETE

Custom Action 5: OPENAPP
- purpose: Open an app.
- format: OPENAPP <APP_NAME>
- example usage: OPENAPP Zoho Meeting

Custom Action 6: WAIT
- purpose: Wait a set number of seconds for something on screen (e.g., a loading bar).
- format: WAIT
- example usage: WAIT

Custom Action 7: LONG_CLICK
- purpose: Long click at the specified position.
- format: LONG_CLICK <point>[[x-axis, y-axis]]</point>
- example usage: LONG_CLICK <point>[[101, 872]]</point>

In most cases, task instructions are high-level and abstract. Carefully read the instruction and action history, then perform reasoning to determine the most appropriate next action. Ensure you strictly generate two sections: Thought and Action.
Thought: Clearly outline your reasoning process for current step.
Action: Specify the actual actions you will take based on your reasoning. You should follow action format above when generating.

And your final goal, previous actions and associated screenshot are as follows:

Final goal: {finalGoal}
Previous actions: {previousActions}
__LOW_LEVEL_PLACEHOLDER__
Screenshot: <image>

Instructions for Determining the Next Action
- Carefully analyze the final goal, previous actions, and the current screenshot.
- Identify the most suitable action based on the context and the goal.
- Make sure the action you suggest aligns with the desired outcome, considering the previous steps.

Figure 8. Prompt template for OS-Atlas.

Prompt Template for AgentCPM-GUI

```
# Role
你是一名熟悉安卓系统触屏GUI操作的智能体，将根据用户的问题，分析当前界面的GUI元素和布局，生成相应的操作。
# Task
针对用户问题，根据输入的当前屏幕截图，输出下一步的操作。
# Rule
- 以紧凑JSON格式输出
- 输出操作必须遵循Schema约束
# Schema
{
  "type": "object",
  "description": "执行操作并决定当前任务状态",
  "additionalProperties": false,
  "properties": {
    "thought": {
      "type": "string",
      "description": "智能体的思维过程"
    },
    "POINT": {
      "$ref": "#/$defs/Location",
      "description": "点击屏幕上的指定位置"
    },
    "to": {
      "description": "移动，组合手势参数",
      "oneOf": [
        {
          "enum": [
            "up",
            "down",
            "left",
            "right"
          ],
          "description": "从当前点（POINT）出发，执行滑动手势操作，方向包括向上、向下、向左、向右"
        },
        {
          "$ref": "#/$defs/Location",
          "description": "移动到某个位置"
        }
      ]
    },
    "duration": {
      "type": "integer",
      "description": "动作执行的时间或等待时间，毫秒",
      "minimum": 0,
      "default": 200
    },
    "PRESS": {
      "type": "string",
      "description": "触发特殊按键，HOME为回到主页按钮，BACK为返回按钮，ENTER为回车按钮",
      "enum": [
        "HOME",
        "BACK",
        "ENTER"
      ]
    },
    "TYPE": {
      "type": "string",
      "description": "输入文本"
    },
    "STATUS": {
      "type": "string",
      "description": "当前任务的状态。特殊情况：satisfied，无需操作；impossible，任务无法完成；interrupt，任务中断；need_feedback，需要用户反馈；",
      "enum": [
        "continue",
        "finish",
        "satisfied",
        "impossible",
        "interrupt",
        "need_feedback"
      ],
      "default": "continue"
    }
  },
  "$defs": {
    "Location": {
      "type": "array",
      "description": "坐标为相对于屏幕左上角位原点的相对位置，并且按照宽高比例缩放到0～1000，数组第一个元素为横坐标x，第二个元素为纵坐标y",
      "items": {
        "type": "integer",
        "minimum": 0,
        "maximum": 1000
      },
      "minItems": 2,
      "maxItems": 2
    }
  }
}
<Question>{instruction}</Question>
```

Figure 9. Prompt template for AgentCPM-GUI.

Prompt Template for GUI-Owl

The user query: {instruction}
Task progress (You have done the following operation on the current device): {history}.
Before answering, explain your reasoning step-by-step in {think_tag_begin}{think_tag_end} tags, and insert them before the <tool_call></tool_call> XML tags.
After answering, summarize your action in <conclusion></conclusion> tags, and insert them after the <tool_call></tool_call> XML tags.
<tools>
{"type": "function", "function": {"name": "mobile_use", "description": "Use a touchscreen to interact with a mobile device, and take screenshots.
* This is an interface to a mobile device with touchscreen. You can perform actions like clicking, typing, swiping, etc.
* Some applications may take time to start or process actions, so you may need to wait and take successive screenshots to see the results of your actions.
* The screen's resolution is '412'x'732'.
* Make sure to click any buttons, links, icons, etc with the cursor tip in the center of the element. Don't click boxes on their edges unless asked.",
"parameters": {"properties": {"action": {"description": "The action to perform. The available actions are:
* `key`: Perform a key event on the mobile device.
    - This supports adb's `keyevent` syntax.
    - Examples: "volume_up", "volume_down", "power", "camera", "clear".
* `click`: Click the point on the screen with coordinate (x, y).
* `long_press`: Press the point on the screen with coordinate (x, y) for specified seconds.
* `swipe`: Swipe from the starting point with coordinate (x, y) to the end point with coordinates2 (x2, y2).
* `type`: Input the specified text into the activated input box.
* `system_button`: Press the system button.
* `open`: Open an app on the device.
* `wait`: Wait specified seconds for the change to happen.
* `terminate`: Terminate the current task and report its completion status.", "enum": ["key", "click", "long_press", "swipe", "type", "system_button", "open", "wait", "terminate"], "type": "string"}, "coordinate": {"description": "(x, y): The x (pixels from the left edge) and y (pixels from the top edge) coordinates to interact with. Required only by `action=click`, `action=long_press`, and `action=swipe`.", "type": "array"}, "coordinate2": {"description": "(x2, y2): The end point coordinates. Required only by `action=swipe`.", "type": "array"}, "text": {"description": "Required only by `action=key`, `action=type`, and `action=open`.", "type": "string"}, "time": {"description": "The seconds to wait. Required only by `action=long_press` and `action=wait`.", "type": "number"}, "button": {"description": "Required only by `action=system_button`. Options:
- "Back": return to previous screen
- "Home": go to home screen
- "Menu": open app background menu
- "Enter": press enter key", "enum": ["Back", "Home", "Menu", "Enter"], "type": "string"}, "status": {"description": "The status of the task. Required only by `action=terminate`.", "type": "string", "enum": ["success", "failure"]}}, "required": ["action"], "type": "object"}}}
</tools>

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>

Figure 10. Prompt template for GUI-Owl.

Supplementary Prompt Template for State-focused Prompt Engineering

```
## Supplementary Notes for State Reasoning

When deciding the next  action, always infer the **current state** of the interface from the latest screenshot.

- Identify whether the current screen already satisfies the user's instruction. If so, avoid redundant actions.
- When interacting with elements that represent **binary or toggle states** (e.g., switches, checkboxes, toggles, radio buttons):
  - **Compare** the current toggle state (ON/OFF, enabled/disabled, checked/unchecked) visible in the GUI with the **user's desired state** as specified in the instruction.
  - **Only click or toggle** the element **if** the current state does **not** match the user's desired state.
  - If the current state already matches the target state, **do nothing** and continue reasoning for the next relevant step.
- When multiple toggles or similar controls exist, carefully identify which one matches the user instruction before performing an action.
- If the toggle state is visually ambiguous, reason conservatively and take a minimal action (e.g., wait or inspect further) before interacting.

Always ensure actions are **state-aware**, **goal-directed**, and avoid unnecessary state changes.
```

Figure 11. Supplementary prompt template for state-focused prompt engineering.

Supplementary Prompt Template for StaR-style Prompting

```
## Supplementary Notes for Toggle State Reasoning

- This reasoning procedure applies **only when the user instruction explicitly involves a GUI toggle, switch, or checkbox** (i.e., an element with ON/OFF or checked/unchecked states).
  For all other types of actions (e.g., button clicks, typing, scrolling), follow the normal reasoning flow without this comparison logic.

- When the instruction targets a toggle element:
  - First, **infer the current state** of the toggle from the screenshot.
  - Then, **compare** it with the **desired state** implied by the user goal.
  - If multiple toggles or similar controls exist, carefully identify which one matches the user instruction before acting.

- Always ensure actions are **state-aware**, **goal-directed**, and avoid unnecessary toggles.

- Follow this explicit reasoning pattern when handling toggle-related instructions:
  - Example reasoning template:
    ```
    According to the screenshot, the switch `{feature}` is currently `{before_state}`, while according to the goal, it should be `{desired_state}`.
    ```
  - If `{before_state}` == `{desired_state}`:
    ```
    No toggle needed. Executing: `set the task as completed`.
    ```
  - Else:
    ```
    Switch needs to be toggled. Executing: `click on the {feature}`.
    ```
```

Figure 12. Supplementary prompt template for StaR-style prompting.

Ground Truth Toggle State Prompting

Note
The target toggle is currently {state} on the screen.

Figure 13. Supplementary prompt template for Ground Truth Toggle State Prompting.