A VERIFIABLE DATA FLYWHEEL FOR MOBILE GUI AGENTS

Xingjian Diao
Department of Computer Science
Dartmouth College
Hanover, NH, USA

ABSTRACT

Mobile GUI agents execute natural-language instructions through sequences of interface actions, but long-horizon execution remains vulnerable to silent transition failures, state drift, and ineffective recovery. A plausible action may fail to produce the intended state transition, causing subsequent decisions to be conditioned on a state that was never reached and allowing these errors to compound over time. Existing systems often address such failures through online reflection or auxiliary verification, introducing additional reasoning and model calls during execution. We introduce TRAVEC, a framework that moves transition prediction, verification, and recovery from the deployment-time execution loop into training. The key abstraction is an evidence-grounded *transition contract*, which specifies the task-relevant state change expected after an action, the observable evidence used to verify that change, and an optional recovery conditioned on transition failure. From heterogeneous mobile GUI trajectories, we construct 550K verifiable supervision items and use them to train a Contract Policy and a Verifier that audit self-generated trajectories and retain audited training steps. The resulting audited experience is then used to optimize the Action Policy through supervised learning and offline GRPO. These auxiliary components are used only during training; deployment retains a single action-only policy with no additional verification or free-form reasoning. Across four offline and two online benchmarks, TRAVEC achieves strong performance across tasks while retaining low inference latency, suggesting that transition-aware supervision can be incorporated into policy learning while retaining action-only deployment.

1 INTRODUCTION

Graphical user interfaces (GUIs) are the primary means through which people access mobile applications, desktop software, and web services, making them a natural environment for agents that translate natural-language instructions into digital actions (Wang et al., 2024b; Tang et al., 2025). In particular, mobile GUI agents execute multi-step tasks through repeated cycles of interface perception, planning, and action generation (Tang et al., 2025; Zhou et al., 2026a). However, a plausible action is not necessarily a successful one: it may be correctly formed and targeted yet fail to induce the intended state transition because of overlays, focus shifts, or delayed interface updates. If the agent proceeds as though the action succeeded, subsequent decisions are conditioned on a state that was never reached. Figure 1 illustrates three representative failure modes in long-horizon mobile GUI interaction: silent transition failures, state drift despite observable changes, and ineffective recovery. Over

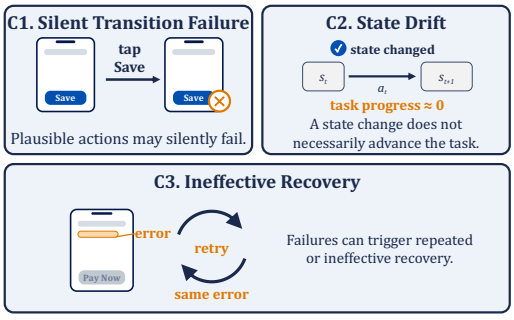


Figure 1: Representative failure modes in long-horizon mobile GUI interaction: (C1) plausible actions can fail silently, (C2) state changes may not indicate task progress, and (C3) recovery attempts may fail to resolve execution errors.

extended trajectories, such local failures can accumulate and ultimately lead to task failure (Rawles et al., 2025; Yu et al., 2026).

Existing GUI-agent supervision is largely organized at two levels. Action-level grounding and imitation data teach a policy what to do and where to act (Hong et al., 2024; Cheng et al., 2024), while trajectory-level environments provide task-level signals of overall task completion (Xie et al., 2024; Rawles et al., 2025; Lai et al., 2026; Liu et al., 2026b). What remains less explicit is the local transition between these two levels: given the current interface and an action, what task-relevant change should occur next, and what evidence would verify that change? Without such a target, a policy may observe either no state change or an unintended one and still continue as if the action had succeeded.

Recent work addresses action consequences through state-transition or world-model pretraining (Liu et al., 2026c; Li et al., 2025b), process rewards and visual state comparison (Chen et al., 2025; Qian, 2026), and state-aware reflection or recovery (Guo et al., 2026b; Zhang et al., 2026; Guo et al., 2026a). However, transition prediction, outcome verification, and recovery are often developed as separate components and, in many systems, remain part of the execution loop, where additional reasoning and model calls increase per-step latency (Huang et al., 2026). This raises a natural question: can these capabilities be unified as training-time supervision and removed from deployment?

We answer this question with an evidence-grounded *transition contract*. Rather than predicting the full next screen, a contract captures only the task-relevant consequence of an action: the current state and preconditions, expected state change, observable evidence, and an optional recovery. Post-action observations serve only as privileged training information for contract construction and validation; they never enter the Action Policy's decision-time input. The contract therefore provides a unified training interface for transition prediction, outcome verification, and failure-conditioned recovery.

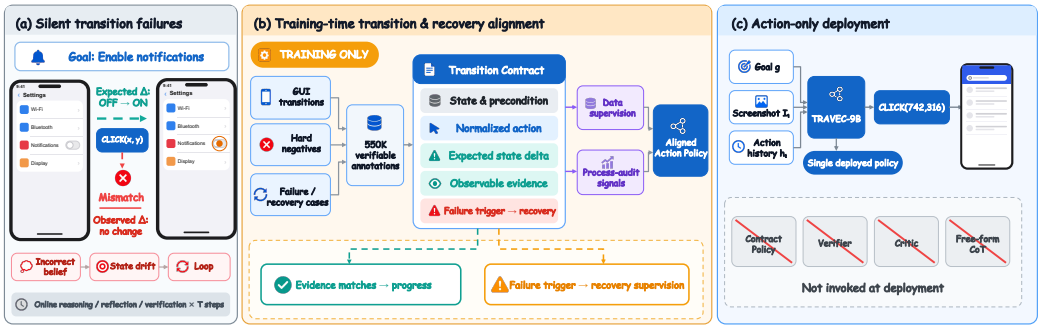


Figure 2: Overview of TRAVEC. Transition prediction, outcome verification, and failure recovery are incorporated into training through evidence-grounded contracts and process-audit signals. Deployment retains only a single action-only policy, with all auxiliary components removed.

Building on this representation, we introduce TRAVEC, a transition-aware training framework for mobile GUI agents, illustrated in Figure 2. We construct 550K verifiable supervision items using deterministic state differencing, evidence-grounded multimodal annotation, counterfactual recovery generation, and model-independent validation. A Contract Policy predicts expected transitions before execution, while a Verifier evaluates realized outcomes afterward; together with a trajectory Critic, they audit self-generated interactions and curate them into training data for policy optimization. The Action Policy is then optimized through supervised learning and offline GRPO with action-alignment and audited process signals. All auxiliary components are used only during training. At deployment, TRAVEC retains a single Action Policy that emits one executable action per step without auxiliary verification or free-form reasoning. Our contributions are threefold:

- We develop a scalable automated pipeline for constructing transition-aware GUI supervision. Using evidence-grounded transition contracts, deterministic state differencing, multimodal annotation, and counterfactual recovery generation, the pipeline produces 550K verifiable supervision items capturing expected state changes, observable evidence, and failure-conditioned recovery.
- We introduce a training-time process-audit framework that combines a Contract Policy, a Verifier, and a trajectory Critic to audit self-generated interactions and curate training data for supervised learning and offline GRPO.

- We train TRAVEC as a standalone action-only policy, using transition verification and recovery only during training. Across six offline and online benchmarks, TRAVEC improves grounding and closed-loop execution while retaining low inference latency without auxiliary verification or free-form reasoning at deployment.

2 Related Work

2.1 GUI Grounding, Planning, Memory, and Recovery

Navigating ambiguous graphical interfaces necessitates robust visual grounding and sequential planning capabilities. Early methodologies primarily optimized spatial resolution and visual-token efficiency to bridge the gap between natural-language instructions and local interface elements (Hong et al., 2024; Cheng et al., 2024; You et al., 2024; Lu et al., 2024; Gou et al., 2025; Wu et al., 2025; Lin et al., 2025). Building upon this perceptual foundation, native action models emerged to support cross-platform action generation and trajectory learning (Xu et al., 2024; Qin et al., 2025; Liu et al., 2026d; Ye et al., 2025; Xu et al., 2026a). To manage the compounding complexity of long-horizon tasks, contemporary architectures frequently incorporate episodic memory, semantic retrieval, and explicit System-2 reflection (Wang et al., 2024a; Li et al., 2024b; Agashe et al., 2025a;b; Qin et al., 2025; Wang et al., 2025; Zhou et al., 2026b; Liu et al., 2026a; Xiao et al., 2026; Lu et al., 2026). However, many of these orchestration mechanisms retain additional memory, reflection, or verification components during execution, which can increase per-step inference cost. TRAVEC instead emphasizes transition-aware supervision that integrates prediction, verification, and recovery within a unified training framework.

2.2 Automated Experience, Reinforcement Learning, and Efficient Execution

The scalability of GUI agents increasingly relies on automated environment exploration and synthetic trajectory generation (Sun et al., 2025; Mu et al., 2025; Cheng et al., 2026). To align these generated experiences with successful task outcomes, recent frameworks apply rule-based reward shaping or partially verifiable Group Relative Policy Optimization offline (Luo et al., 2025; Zhang et al., 2025; Yang et al., 2026; Lu et al., 2025b). Recent systems further expand this learning paradigm by integrating dynamic task generation and post-hoc evidence auditing (Lai et al., 2026; Xu et al., 2026b; Liu et al., 2026b; Li et al., 2026). Recent efforts to reduce inference latency include context compression and anticipated decoding (Huang et al., 2025; 2026; Dong et al., 2026). TRAVEC takes a complementary approach by moving transition prediction, evidence-grounded verification, and recovery supervision into the training pipeline. As a result, deployment retains a single action-only policy without auxiliary verification or free-form reasoning.

3 Method

TRAVEC is built around a central principle: represent each action with an evidence-grounded transition contract, use these contracts to audit collected interaction experience, and distill the audited experience into an action-only policy. We realize this principle through automated transition supervision, role-specialized adapters, and process-audited optimization.

3.1 Problem formulation

We model goal-conditioned GUI interaction as a partially observable decision process. The observation $o_t = (I_t, U_t, X_t)$ comprises a screenshot, available interface structure, and extracted text. During attempt k of task i, the Action Policy conditions on screenshot I_t, goal g, action history $h_t = (a_1, \ldots, a_{t-1})$, and earlier-attempt hints $\eta_i^{(<k)}$:

$$\pi_{\theta_A}(a_t \mid g, I_t, h_t, \eta_i^{(<k)}), \qquad a_t = (\alpha_t, \psi_t). \tag{1}$$

Here α_t and ψ_t denote action type and arguments. We maximize $\mathbb{E}[Y(\tau)]$ for binary task outcome $Y(\tau)$. Deployment requires one valid action per step with empty hints and no auxiliary model calls.

A transition contract specifies expected consequences and recovery:

$$c_t = (\tilde{s}_t, p_t, a_t, \hat{d}_t, E_t, \rho_t). \tag{2}$$

Its components are task state $\tilde{s}_t$, precondition p_t, action a_t, expected change $\hat{d}_t$, observable evidence E_t, and recovery specification ρ_t. Recovery binds a failure trigger to a corrective action, expected change, and evidence. The Contract Policy models $q_{\theta_C}(c_t \mid g, o_t, h_t, a_t)$ before execution. After execution, the Verifier models $V_{\theta_V}(y_t^V \mid g, o_t, c_t, o_{t+1}, d_t, \chi_t)$ using the observed difference $d_t = \mathcal{D}(o_t, o_{t+1})$ and validation indicators χ_t. The output $y_t^V = (\mathbf{v}_t, \nu_t, \kappa_t)$ comprises quality scores, a verdict, and confidence $\kappa_t \in \{0, 1, 2, 3, 4\}$, with

$$\mathbf{v}_t = (v_t^{\text{fmt}}, v_t^{\text{act}}, v_t^{\text{state}}, v_t^{\text{delta}}, v_t^{\text{progress}}, v_t^{\text{recovery}}, v_t^{\text{safe}}, v_t^{\text{eff}}) \in \{0, 1, 2, 3, 4\}^8. \tag{3}$$

These candidate-specific scores measure format validity, action validity, state grounding, delta evidence, task progress, recovery quality, loop safety, and cost efficiency. Recovery requires an active failure trigger; loop safety distinguishes ineffective cycles from justified retries. Higher scores indicate stronger support. Appendix A.1 specifies the contract constraints.

3.2 Automated Data Annotation and Generation

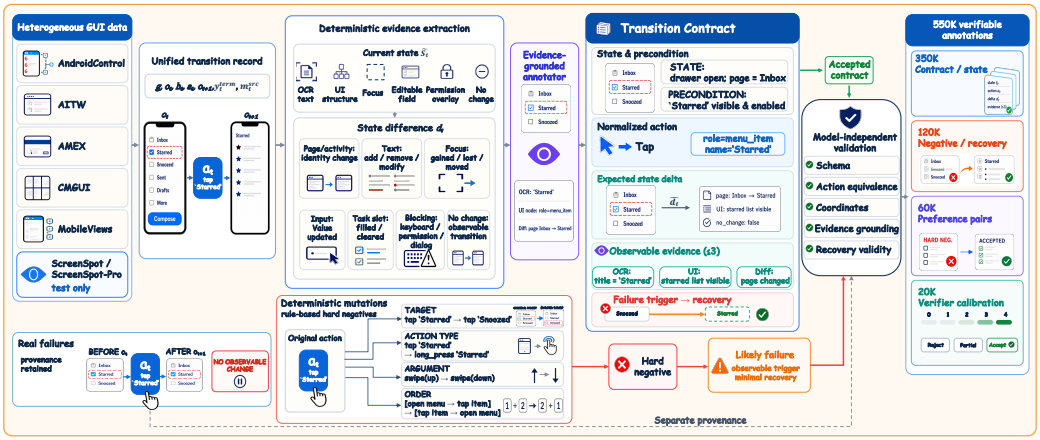


Figure 3: Automated construction of verifiable transition supervision. Deterministic state differencing, evidence-grounded annotation, counterfactual generation, and model-independent validation produce 550K annotations spanning transition contracts, state supervision, hard negatives, failure recovery, preference pairs, and verifier calibration for downstream policy training.

Multimodal labeling and deterministic validation yield 550K annotations, as shown in Figure 3. We unify AndroidControl, AITW, AMEX, CMGUI, MobileViews, ScreenSpot, and ScreenSpot-Pro. ScreenSpot and ScreenSpot-Pro are held out for evaluation and contribute no training examples.

Contract annotation. Deterministic extraction produces a state estimate $\bar{s}_t$ and transition difference d_t for each observed transition x_t. Multimodal annotation then generates a candidate contract $c_t^+ = A_C(x_t, \bar{s}_t, d_t)$. A contract is accepted only if its action is semantically equivalent to the demonstrated action and its structure, evidence, and recovery specification pass validation. State and preconditions are derived from the current observation, while subsequent observations supervise expected changes and evidence without entering the policy input. Static grounding samples provide localization evidence.

Negative, recovery, and calibration supervision. Rule-based action perturbations produce syntactically plausible hard negatives. The annotator predicts their likely failures, observable triggers, and minimal recovery branches. Observed and counterfactual failures retain separate provenance. Preference pairs contrast accepted contracts against negatives under the same decision input. Positive transitions, counterfactual negatives, and observed failures provide verifier calibration targets. Appendix A.2 details the transition representation and annotation procedure.

3.3 Supervised Training of Role-Specialized Adapters

Each role uses a Qwen3.5-9B backbone W_0 with LoRA parameters ϕ, as shown in Figure 4. Action SFT learns ϕ_A^{SFT} by minimizing cross-entropy over normalized action completions with a frozen

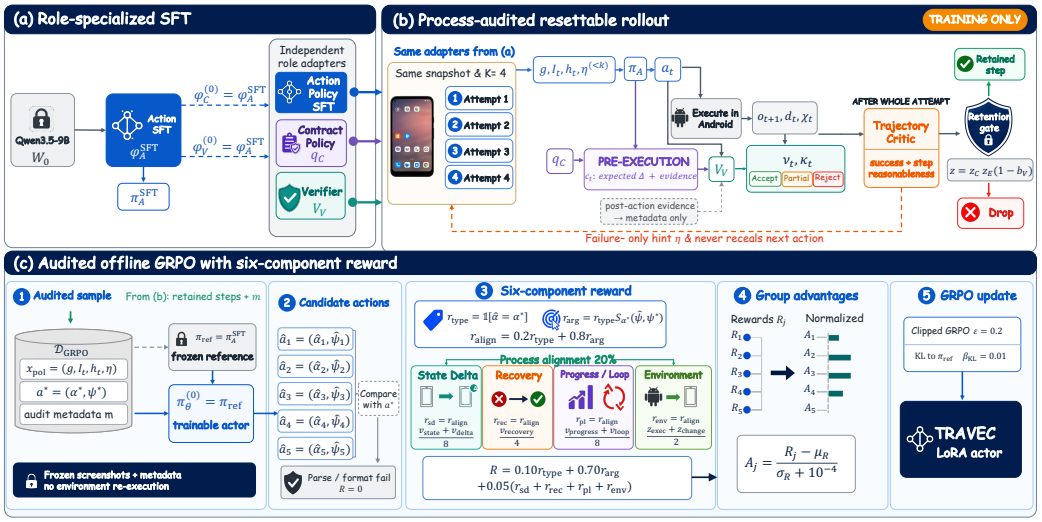


Figure 4: Three-stage policy optimization framework. Role-specialized SFT initializes the Action, Contract, and Verifier policies, resettable rollouts collect and audit interaction experience with transition-aware signals, and offline GRPO optimizes the Action Policy with a six-component reward combining reference alignment and process-grounded signals.

vision encoder. The auxiliary policies initialize from $\phi_C^{(0)} = \phi_V^{(0)} = \phi_A^{\mathrm{SFT}}$ and train independently. The Contract Policy learns contracts and recovery from decision-time information. The Verifier learns post-action assessments from contracts, observed transitions, and validation indicators. Their parameters remain separate during rollout.

3.4 Process-audited offline GRPO

Rollout and audit. We collect $K = 4$ attempts on each of 400 AndroidWorld training tasks, resetting the initial state between attempts. The Action Policy proposes actions, the Contract Policy predicts contracts, and the Verifier assesses transitions. A fixed multimodal trajectory Critic assesses success $\hat{Y}_i^{(k)} \in \{0,1\}$ and step reasonableness $z_{i,t}^{C,(k)} \in \{0,1\}$ from goals, actions, and visual execution evidence.

Step selection combines Critic approval, execution validity, and a veto for high-confidence verifier rejection:

$$z_{i,t}^{(k)} = z_{i,t}^{C,(k)} z_{i,t}^{E,(k)} (1 - b_{i,t}^{V,(k)}), \qquad b_{i,t}^{V,(k)} = \mathbf{1}[\nu_{i,t}^{(k)} = \text{reject} \wedge \kappa_{i,t}^{(k)} \geq 3]. \tag{4}$$

Here z^E indicates successful execution without errors or action invalidity. Critic hints $\eta_i^{(<k)}$ describe prior failures and behavior to avoid. They exclude correct next actions and are initially empty.

Offline sample selection. We retain tasks with Critic-assessed empirical success rates below 1 across the four attempts. For attempt length $T_i^{(k)}$, we define

$$q_i^{(k)} = \frac{\sum_t z_{i,t}^{(k)}}{\max(1, T_i^{(k)})}, \qquad k_i^* = \arg\operatorname*{lex\,max}_k \big(\hat{Y}_i^{(k)}, q_i^{(k)}, -T_i^{(k)}\big). \tag{5}$$

Selection prioritizes success, retained-step fraction, then shorter length. Selected steps form

$$\mathcal{D}_{\mathrm{GRPO}} = \{(x_{i,t}^{\mathrm{pol}}, a_{i,t}^*, m_{i,t}) : z_{i,t}^{(k_i^*)} = 1\}. \tag{6}$$

The fixed input x^{pol} contains the goal, screenshot, action history, and collection-time hints. The reference a^* and audit context m are retained (Appendix A.3).

Six-component reward. Following adaptive action scoring (Liu et al., 2026b), a candidate $\hat{a} = (\hat{\alpha}, \hat{\psi})$ is compared with the retained action $a^* = (\alpha^*, \psi^*)$:

$$r_{\mathrm{type}} = \mathbf{1}[\hat{\alpha} = \alpha^*], \qquad r_{\mathrm{arg}} = r_{\mathrm{type}} S_{\alpha^*}(\hat{\psi}, \psi^*). \tag{7}$$

S_{α^*} measures type-specific argument similarity; invalid completions receive zero reward. Within the shared audit context m, each candidate receives a separate contract and evidence-grounded process assessment:

$$r_{\mathrm{sd}} = (v^{\mathrm{state}} + v^{\mathrm{delta}})/8, \qquad r_{\mathrm{rec}} = v^{\mathrm{recovery}}/4, \tag{8}$$

$$r_{\mathrm{pl}} = (v^{\mathrm{progress}} + v^{\mathrm{safe}})/8, \qquad r_{\mathrm{env}} = (z_{\mathrm{exec}} + z_{\mathrm{change}})/2. \tag{9}$$

All scores are candidate-specific. State Delta assesses grounded preconditions and transition evidence. Recovery measures correction of a pre-existing failure and is zero without an active trigger. Progress/Loop assesses unfinished requirements and avoidance of evidenced ineffective cycles, allowing justified retries, backtracking, and waiting.

The binary indicators z_{exec} and z_{change} require candidate-attributable execution success and task-relevant outcome evidence under the same decision state. Unrelated changes earn no outcome credit; waiting and termination need not change the screen.

The verifier-derived process rewards are normalized to $[0, 1]$, while r_{env} is the average of two binary indicators. We combine them as

$$R = w_{\mathrm{type}} r_{\mathrm{type}} + w_{\mathrm{arg}} r_{\mathrm{arg}} + w_{\mathrm{sd}} r_{\mathrm{sd}} + w_{\mathrm{rec}} r_{\mathrm{rec}} + w_{\mathrm{pl}} r_{\mathrm{pl}} + w_{\mathrm{env}} r_{\mathrm{env}}. \tag{10}$$

Reference similarity affects only the first two terms; process credit can also be assigned to alternative actions supported by execution evidence. Appendices A.4 and B.2 specify scoring rules and weights.

Optimization and deployment. The actor and fixed reference are initialized from Action SFT. For each fixed input, the sampling policy $\pi_{\theta_{\mathrm{old}}}$ generates $G = 5$ completions. Each completion receives reward $R_j = R(\hat{a}_j, a^*, m)$ and advantage $A_j = (R_j - \mu_R)/(\sigma_R + 10^{-4})$, where μ_R and σ_R are the mean and population standard deviation of the G rewards. With clipping radius $\epsilon = 0.2$ and KL coefficient $\beta_{\mathrm{KL}} = 0.01$, we minimize

$$\mathcal{L}_{\mathrm{GRPO}} = -\mathbb{E}\left[\frac{1}{G}\sum_{j=1}^{G}\frac{1}{L_j}\sum_{\ell=1}^{L_j}\Big(\min\{\xi_{j,\ell}A_j, \mathrm{clip}(\xi_{j,\ell}, 1-\epsilon, 1+\epsilon)A_j\} - \beta_{\mathrm{KL}} D_{\mathrm{KL}}^{j,\ell}\Big)\right], \tag{11}$$

where L_j is completion length and $D_{\mathrm{KL}}^{j,\ell}$ regularizes against the fixed SFT reference. The token probability ratio is

$$\xi_{j,\ell} = \frac{\pi_\theta(\hat{a}_{j,\ell} \mid x, \hat{a}_{j,<\ell})}{\pi_{\theta_{\mathrm{old}}}(\hat{a}_{j,\ell} \mid x, \hat{a}_{j,<\ell})}. \tag{12}$$

Collection alternates with offline optimization. Each round executes candidates separately from the same decision-state snapshot, recording successor observations, execution outcomes, and verifier scores. GRPO freezes candidates and evidence, using only complete records; new candidates enter after the next collection round (Appendix A.5). Deployment uses only $\pi_{\theta_A^{\mathrm{TRAVEC}}}(a_t \mid g, I_t, h_t, \emptyset)$, which emits one action per step.

4 Experiments

4.1 Experimental setup

We initialize TRAVEC from the post-trained Qwen3.5-9B multimodal model. Action, Contract, and Verifier SFT use rank-16 LoRA, while GRPO uses rank-64 LoRA. The vision encoder remains frozen throughout training. Additional optimization details are provided in Appendix B.1.

Benchmarks and metrics. We evaluate TRAVEC-9B on four offline and two online benchmarks. The offline benchmarks include AndroidControl (Li et al., 2024a), GUI-Odyssey (Lu et al., 2025a), ScreenSpot (Cheng et al., 2024), and ScreenSpot-Pro (SS-Pro) (Li et al., 2025a), while the online benchmarks are AndroidWorld (Rawles et al., 2025) and MobileWorld (Kong et al., 2026). We report action accuracy under high-level and low-level instructions on AndroidControl, type match, grounding rate, and step success rate on GUI-Odyssey, target-hit accuracy on ScreenSpot and ScreenSpot-Pro, episode success rate on AndroidWorld, and pass-at-one success rate on MobileWorld. Benchmark and evaluation details are provided in Appendix B.3.

Comparison models. Qwen3.5-9B serves as our multimodal base model (Qwen Team, 2026). We compare TRAVEC-9B against seven publicly available 7B–8B GUI agents: Aguvis-7B-720P (Xu

et al., 2024), GUI-R1-7B (Luo et al., 2025), ScaleCUA-7B (Liu et al., 2026d), GUI-Libra-8B (Yang et al., 2026), OpenMobile-8B (Cheng et al., 2026), MemGUI-8B-SFT (Liu et al., 2026a), and ForgeQwen3-8B (Liu et al., 2026b). Detailed descriptions are provided in Appendix B.4.

4.2 Data composition and sampling

Figure 5 summarizes the composition of our 550,000 verifiable supervision items, their data sources, and the hard-case sampling distribution. The corpus consists of 63.64% transition-contract and state items, 21.82% negative-recovery items, 10.91% preference pairs, and 3.64% verifier-calibration items. Training supervision is constructed from AITW, the training split of AndroidControl, CMGUI, and AMEX, while MobileViews is used only for parser construction. ScreenSpot and ScreenSpot-Pro are reserved exclusively for evaluation and contribute no training examples.

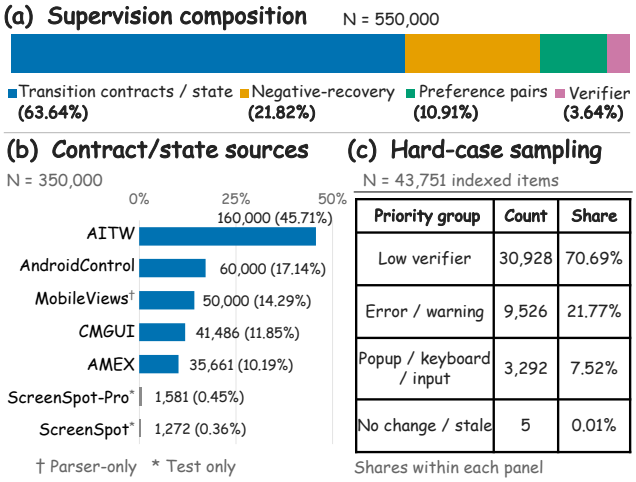


Figure 5: Supervision composition, data sources, and hard-case sampling statistics. Parser-only and evaluation-only sources are explicitly marked in panel (b).

Train–test separation. We maintain benchmark-specific separation between training and evaluation data. AndroidControl supervision is constructed only from its training split, while all reported results use the held-out test split. The 400 AndroidWorld tasks used for resettable rollout are disjoint from the task templates used for evaluation, and MobileWorld is used only for evaluation. No evaluation examples are used for SFT, hard-case sampling, or GRPO.

To increase exposure to challenging transitions during training, we identify 43,751 hard-case items. Among them, 70.69% correspond to low-verifier-confidence cases, 21.77% to environment errors or warnings, and the remainder to popup, keyboard, input, or stale-state cases. These samples are upsampled during policy optimization to emphasize difficult transitions and execution failures.

4.3 Offline benchmark results

Table 1: Comparison with baseline GUI agents across offline benchmarks and inference efficiency. High/Low denote action accuracy on AndroidControl under high-/low-level instructions. TM, GR, and SR denote type match, grounding rate, and step success rate on GUI-Odyssey. HT denotes target-hit accuracy on ScreenSpot and ScreenSpot-Pro (SS-Pro). Lat. and Tok. denote mean inference time and generated tokens per action. All models are evaluated under the same hardware and benchmark settings. Best results are shown in **bold**, and second-best results are underlined.

Model	AndroidControl		GUI-Odyssey			ScreenSpot	SS-Pro	Efficiency	
	High ↑	Low ↑	TM ↑	GR ↑	SR ↑	HT ↑	HT ↑	Lat. ↓	Tok. ↓
Aguvis-7B-720P	57.237	66.914	74.137	70.541	53.373	78.852	20.936	2.853	34.505
GUI-R1-7B	22.464	42.436	70.889	58.397	42.590	85.849	28.716	6.224	32.096
ScaleCUA-7B	36.056	46.608	77.218	69.886	57.315	73.899	23.340	3.591	35.800
GUI-Libra-8B	62.856	81.417	63.181	71.155	43.994	85.535	53.700	5.905	78.294
OpenMobile-8B	52.335	72.502	67.164	59.893	42.332	79.324	38.646	5.763	58.644
MemGUI-8B-SFT	58.061	77.424	68.932	62.140	45.444	76.965	39.216	12.910	318.260
ForgeQwen3-8B	62.665	79.850	**79.864**	71.221	58.369	82.704	46.490	7.070	86.799
Qwen3.5-9B	34.379	52.966	62.217	90.823	56.185	38.758	42.631	0.815	18.039
TRAVEC-9B (ours)	**64.309**	**93.337**	79.588	**91.006**	**72.212**	**92.689**	**67.805**	**0.786**	**17.152**

Table 1 shows that TRAVEC-9B ranks first on six of seven offline task metrics and second on the remaining one. On AndroidControl, TRAVEC-9B reaches 64.309% and 93.337% accuracy under High- and Low-level instructions, improving over Qwen3.5-9B by 29.930 and 40.371 percentage points and exceeding the strongest specialized baselines by 1.453 and 11.920 points, respectively. On GUI-Odyssey, TRAVEC-9B reaches 79.588% type match, 91.006% grounding rate, and 72.212% step success. Relative to Qwen3.5-9B, these correspond to gains of 17.371, 0.183, and 16.027 points. It trails ForgeQwen3-8B by 0.276 points in type match while exceeding the best specialized grounding and step-success results by 19.785 and 13.843 points.

TRAVEC-9B also achieves the best target-hit accuracy on both grounding benchmarks while retaining the lowest measured inference cost in the comparison. On ScreenSpot and ScreenSpot-Pro, it reaches 92.689% and 67.805%, improving over Qwen3.5-9B by 53.931 and 25.174 percentage points and over the strongest specialized baselines by 6.840 and 14.105 points, respectively. Under the same hardware and benchmark settings, TRAVEC-9B requires 0.786 seconds and 17.152 generated tokens per action, compared with 0.815 seconds and 18.039 tokens for Qwen3.5-9B. Aguvis-7B-720P is the fastest specialized baseline at 2.853 seconds per action, corresponding to a 72.4% lower mean inference latency for TRAVEC-9B under this evaluation setup.

4.4 Online benchmark results

Table 2 shows that TRAVEC-9B achieves the highest success rate among the compared systems on both online benchmarks. On AndroidWorld, TRAVEC-9B reaches 35.345% episode success, improving over Qwen3.5-9B by 14.655 percentage points and exceeding the strongest specialized baseline, ForgeQwen3-8B, by 0.862 points. On MobileWorld, TRAVEC-9B achieves 15.385% pass-at-one success, improving over Qwen3.5-9B by 10.257 points and surpassing the strongest comparison baseline, OpenMobile-8B, by 6.838 points. Thus, TRAVEC-9B ranks first among the compared systems under both online evaluation protocols.

Table 2: Comparison with baseline GUI agents on online benchmarks. AndroidWorld uses episode success rate, whereas MobileWorld uses pass-at-one success rate. Best and second-best results are shown in **bold** and underlined.

Model	AndroidWorld	MobileWorld
	SR ↑	SR ↑
Aguvis-7B-720P	9.483	1.709
GUI-R1-7B	12.931	3.419
ScaleCUA-7B	14.655	3.419
GUI-Libra-8B	19.828	5.983
OpenMobile-8B	18.103	<u>8.547</u>
MemGUI-8B-SFT	25.000	4.273
ForgeQwen3-8B	<u>34.483</u>	5.983
Qwen3.5-9B	20.690	5.128
TRAVEC-9B (ours)	**35.345**	**15.385**

The online benchmarks complement the offline evaluation by measuring closed-loop task completion in interactive environments rather than action prediction or grounding in isolation. AndroidWorld evaluates episode-level success in a dynamic Android environment, while MobileWorld uses a distinct pass-at-one protocol for interactive mobile tasks. The results on both benchmarks therefore complement the offline findings with evidence from multi-step interactive execution.

4.5 Ablating training components

Table 3 shows that removing individual training components reduces performance across the offline benchmarks, with Action SFT having the largest overall effect. Without Action SFT, AndroidControl High/Low drop by 14.856 and 19.167 points, while ScreenSpot-Pro decreases by 11.195 points. Removing the Contract Policy or its explicit fields also produces clear reductions on AndroidControl and ScreenSpot-Pro, whereas changes on ScreenSpot are comparatively small.

The AndroidWorld ablations show larger differences among the training components under closed-loop execution. Removing Action SFT reduces success from 35.345% to 14.080%, followed by removing recovery supervision at 17.816%, the trajectory Critic at 20.977%, and the Transition Verifier at 22.414%. Removing the Contract Policy yields 27.011%, whereas removing only its explicit fields has a smaller effect at 32.759%. Together, these results indicate that multiple components contribute to TRAVEC-9B, with the largest online drops observed when Action SFT, recovery supervision, Critic-based filtering, or verification is removed.

Table 3: Component ablations across offline and online benchmarks. We remove individual training components from TRAVEC-9B and report performance on AndroidControl, ScreenSpot, ScreenSpot-Pro, and AndroidWorld. Best and second-best results are shown in **bold** and underlined.

Model	AndroidControl		ScreenSpot	SS-Pro	AndroidWorld
	High ↑	Low ↑	HT ↑	HT ↑	SR ↑
w/o Critic	58.048	82.097	92.452	63.061	20.977
w/o Contract	57.024	80.110	<u>92.610</u>	63.820	27.011
w/o Verifier	<u>59.573</u>	82.994	92.374	63.947	22.414
w/o Contract Fields	56.344	82.816	91.824	62.998	<u>32.759</u>
w/o Recovery	56.723	<u>86.553</u>	92.296	<u>64.643</u>	17.816
w/o Action SFT	49.453	74.170	89.937	56.610	14.080
TRAVEC-9B	**64.309**	**93.337**	**92.689**	**67.805**	**35.345**

4.6 Ablating reward composition

Table 4: Reward-composition ablation. Weights are ordered as Type, Argument, State Delta, Recovery, Progress, and Environment. The final row reports the configuration adopted by TRAVEC-9B. Best and second-best results are shown in **bold** and underlined.

Reward weights	AndroidControl		ScreenSpot	SS-Pro
	High ↑	Low ↑	HT ↑	HT ↑
$0.125, 0.875, 0, 0, 0, 0$	54.890	78.816	92.060	62.998
$0.0625, 0.4375, 0.125, 0.125, 0.125, 0.125$	<u>56.523</u>	82.425	92.453	<u>63.820</u>
$0.025, 0.175, 0.2, 0.2, 0.2, 0.2$	55.631	<u>82.745</u>	**92.846**	63.631
$0, 0, 0.25, 0.25, 0.25, 0.25$	55.952	81.152	92.060	63.188
$0.10, 0.70, 0.05, 0.05, 0.05, 0.05$	**64.309**	**93.337**	<u>92.689</u>	**67.805**

Table 4 compares five reward compositions spanning action-only, process-only, and mixed objectives. Across the tested configurations, performance does not vary monotonically with the total weight assigned to process-based rewards. The adopted configuration achieves the best results on AndroidControl High, AndroidControl Low, and ScreenSpot-Pro at 64.309%, 93.337%, and 67.805%, respectively, while ranking second on ScreenSpot at 92.689%. In comparison, the action-only configuration is lower by 9.419, 14.521, and 4.807 points on the three metrics where the adopted setting ranks first, while the process-only configuration is lower by 8.357, 12.185, and 4.617 points.

The intermediate mixtures show that the preferred weighting varies somewhat across benchmarks. In particular, the configuration assigning 80% of its weight to process-based terms achieves the highest ScreenSpot result at 92.846%, exceeding the adopted setting by 0.157 points, but performs worse on AndroidControl and ScreenSpot-Pro. Among the five tested configurations, the adopted 80% action-matching and 20% process-based weighting ranks first on three metrics and second on the remaining metric. We therefore use this configuration in TRAVEC-9B.

5 Conclusion

We introduced TRAVEC, a training framework for mobile GUI agents built around evidence-grounded transition contracts that capture expected state changes, observable evidence, and failure-conditioned recovery. These contracts support training-time transition prediction and post-action verification, while auxiliary policies and a trajectory Critic audit self-generated interaction experience for policy optimization. The resulting TRAVEC-9B is deployed as a single action-only policy without auxiliary verification, Critic calls, or free-form reasoning. Across four offline and two online benchmarks, TRAVEC-9B ranks first on six of seven offline task metrics and on both online benchmarks, while retaining the lowest measured inference latency in our comparison. Ablations further indicate that both the training design and reward composition affect performance across the evaluated settings. Overall, these results suggest that transition-aware supervision can be integrated into policy learning while retaining action-only deployment.

AI USE STATEMENT

Generative AI tools were used only to improve the writing, grammar, and presentation of this manuscript. All research ideas, methods, experiments, analyses, and conclusions were developed and verified by the authors.

ETHICS STATEMENT

This work uses existing research datasets and controlled benchmark environments. We do not collect additional personal user data or conduct new studies involving human participants.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, training procedure, data construction pipeline, reward formulation, evaluation protocols, and hyperparameter configurations in the main paper and appendix. These details are intended to enable faithful reproduction of our training and evaluation setup. Upon acceptance, we will release the code and constructed dataset associated with this work.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Wang. Agent s: An open agentic framework that uses computers like a human. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/pdf/2410.08164.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025b. URL https://arxiv.org/pdf/2504.00906.

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, et al. Gui-shepherd: Reliable process reward and verification for long-sequence gui tasks. *arXiv preprint arXiv:2509.23738*, 2025. URL https://arxiv.org/pdf/2509.23738.

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 *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.505.pdf.

Kanzhi Cheng, Zehao Li, Zheng Ma, Nuo Chen, Jialin Cao, Qiushi Sun, Zichen Ding, Fangzhi Xu, Hang Yan, Jiajun Chen, et al. Openmobile: Building open mobile agents with task and trajectory synthesis. *arXiv preprint arXiv:2604.15093*, 2026. URL https://arxiv.org/pdf/2604.15093.

Zihan Dong, Rui Qian, Qishi Zhan, Dongshen Peng, Kaixin Li, and Yu Li. Why are gui agents correct but late? decode on the decision-time critical path, tested with pre-compiled policy trees. *arXiv preprint arXiv:2607.28399*, 2026. URL https://arxiv.org/pdf/2607.28399.

Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/pdf/2410.05243.

Linqiang Guo, Wei Liu, Li Gu, Yang Wang, et al. Stepreflect: Structured ui transition reflection for mobile gui agents. *arXiv preprint arXiv:2608.05587*, 2026a. URL https://arxiv.org/pdf/2608.05587.

Linqiang Guo, Wei Liu, Yi Wen Heng, Tse-Hsun Peter Chen, and Yang Wang. Agent-sama: state-aware mobile assistant. In *AAAI Conference on Artificial Intelligence*, 2026b. URL https://arxiv.org/pdf/2505.23596.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/pdf/2312.08914.

Kung-Hsiang Huang, Haoyi Qiu, Yutong Dai, Caiming Xiong, and Chien-Sheng Wu. Gui-kv: Efficient gui agents via kv cache with spatio-temporal awareness. *arXiv preprint arXiv:2510.00536*, 2025. URL https://arxiv.org/pdf/2510.00536.

Runxi Huang, Liyu Zhang, Shengzhong Liu, and Xiaomin Ouyang. Mobileexplorer: Accelerating on-device inference for mobile gui agents via online exploration. *arXiv preprint arXiv:2605.26546*, 2026. URL https://arxiv.org/pdf/2605.26546.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. In *Annual Meeting of the Association for Computational Linguistics*, 2026. URL https://aclanthology.org/2026.acl-long.278.pdf.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/pdf/2508.14040.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *International Conference on Multimedia*, 2025a. URL https://arxiv.org/pdf/2504.07981.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. Mobileworldbench: Towards semantic world modeling for mobile agents. *arXiv preprint arXiv:2512.14014*, 2025b. URL https://arxiv.org/pdf/2512.14014.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. In *Advances in Neural Information Processing Systems*, 2024a. URL https://arxiv.org/pdf/2406.03679.

Yanda Li, Chi Zhang, Wenjia Jiang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*, 2024b. URL https://arxiv.org/pdf/2408.11824.

Zehao Li, Zhenyu Wu, Yibo Zhao, Bowen Yang, Jingjing Xie, Zhaoyang Liu, Zhoumianze Liu, Kaiming Jin, Jianze Liang, Zonglin Li, et al. Os-themis: A scalable critic framework for generalist gui rewards. *arXiv preprint arXiv:2603.19191*, 2026. URL https://arxiv.org/pdf/2603.19191.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/pdf/2411.17465.

Guangyi Liu, Gao Wu, Congxiao Liu, Pengxiang Zhao, Liang Liu, Mading Li, Qi Zhang, Mengyan Wang, Liang Guo, and Yong Liu. Memgui-agent: An end-to-end long-horizon mobile gui agent with proactive context management. *arXiv preprint arXiv:2606.19926*, 2026a. URL https://arxiv.org/pdf/2606.19926.

Guangyi Liu, Pengxiang Zhao, Gao Wu, Yiwen Yin, Mading Li, Liang Liu, Congxiao Liu, Zhang Qi, Mengyan Wang, Liang Guo, et al. Mobileforge: Annotation-free adaptation for mobile gui agents with hierarchical feedback-guided policy optimization. *arXiv preprint arXiv:2606.19930*, 2026b. URL https://arxiv.org/pdf/2606.19930.

Xiangyan Liu, Kaixin Li, Haonan Wang, Biao Wu, Meng Fang, Longxu Dou, Chao Du, Michael Qizhe Shieh, and Tianyu Pang. Scaling gui agents with visual state transitions. *arXiv preprint arXiv:2607.24112*, 2026c. URL https://arxiv.org/pdf/2607.24112.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. In *International Conference on Learning Representations*, 2026d. URL https://arxiv.org/pdf/2509.15221.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *International Conference on Computer Vision*, 2025a. URL https://arxiv.org/pdf/2406.08451.

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/pdf/2408.00203.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, et al. Ui-s1: Advancing gui automation via semi-online reinforcement learning. *arXiv preprint arXiv:2509.11543*, 2025b. URL https://arxiv.org/pdf/2509.11543.

Zhengxi Lu, Fei Tang, Guangyi Liu, Jin Ma, Kaitao Song, Xu Tan, Wenqi Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Ui-copilot: Advancing long-horizon gui automation via tool-integrated policy optimization. In *Annual Meeting of the Association for Computational Linguistics*, 2026. URL https://aclanthology.org/2026.acl-long.904.pdf.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025. URL https://arxiv.org/pdf/2504.10458.

Jian Mu, Chaoyun Zhang, Chiming Ni, Lu Wang, Bo Qiao, Kartik Mathur, Qianhui Wu, Yuhang Xie, Xiaojun Ma, Mengyu Zhou, et al. GUI-360°: A comprehensive dataset and benchmark for computer-using agents. *arXiv preprint arXiv:2511.04307*, 2025. URL https://arxiv.org/pdf/2511.04307.

Jiachen Qian. Viscritic: Visual state comparison as process reward for gui agents. *arXiv preprint arXiv:2606.24525*, 2026. URL https://arxiv.org/pdf/2606.24525.

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. URL https://arxiv.org/pdf/2501.12326.

Qwen Team. Qwen3.5-9b. Hugging Face model repository, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B. Accessed 2026-08-26.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/pdf/2405.14573.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. In *Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://aclanthology.org/2025.acl-long.277.pdf.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, et al. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*, 2025. URL https://arxiv.org/abs/2504.13865.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025. URL https://arxiv.org/pdf/2509.02544.

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*, 2024a. URL https://arxiv.org/pdf/2406.01014.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, et al. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*, 2024b. URL https://arxiv.org/abs/2411.04890.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: Foundation action model for generalist gui agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/pdf/2410.23218.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. Ui-mem: Self-evolving experience memory for online reinforcement learning in mobile gui agents. *arXiv preprint arXiv:2602.05832*, 2026. URL https://arxiv.org/pdf/2602.05832.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/pdf/2404.07972.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026a. URL https://arxiv.org/pdf/2602.16855.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Jiayu Huang, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Yuxiao Dong, et al. Mobilerl: Online agentic reinforcement learning for mobile gui agents. In *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/pdf/2509.18119.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024. URL https://arxiv.org/pdf/2412.04454.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baolin Peng, Huan Zhang, Jianfeng Gao, et al. Gui-libra: Training native gui agents to reason and act with action-aware supervision and partially verifiable rl. *arXiv preprint arXiv:2602.22190*, 2026. URL https://arxiv.org/pdf/2602.22190.

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: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025. URL https://arxiv.org/pdf/2508.15144.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *European Conference on Computer Vision*, 2024. URL https://arxiv.org/pdf/2404.05719.

Shengcheng Yu, Yuchen Ling, Junyang Xing, Quan Zhou, Chunrong Fang, and Zhenyu Chen. Software engineering for and with gui agent. *arXiv preprint arXiv:2608.09278*, 2026. URL https://arxiv.org/pdf/2608.09278.

Yuzhe Zhang, Xianwei Xue, Xingyong Wu, Mengke Chen, Chen Liu, Xinran He, Run Shao, Feiran Liu, Huanmin Xu, Qiutong Pan, et al. Don't act blindly: Robust gui automation via action-effect verification and self-correction. In *Annual Meeting of the Association for Computational Linguistics*, 2026. URL https://aclanthology.org/2026.acl-long.1335.pdf.

Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, et al. Agentcpm-gui: Building mobile-use agents with reinforcement fine-tuning. In *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2025. URL https://aclanthology.org/2025.emnlp-demos.12.pdf.

Hanzhang Zhou, Panrong Tong, Xu Zhang, Quyu Kong, Chenglin Cai, Tianyu Xia, Gongjie Zhang, Jianan Zhang, Long Li, Long Chen, et al. Qwen-ui-agent technical report: Toward next-generation real-world centric foundation gui agents. *arXiv preprint arXiv:2607.28227*, 2026a. URL https://arxiv.org/pdf/2607.28227.

Xurui Zhou, Gongwei Chen, Yuquan Xie, Zaijing Li, Kaiwen Zhou, Shuai Wang, Shuo Yang, Zhuotao Tian, and Rui Shao. Hiconagent: History context-aware policy optimization for gui agents. In *Conference on Computer Vision and Pattern Recognition*, 2026b. URL https://arxiv.org/pdf/2512.01763.

CONTENTS OF APPENDIX

A SUPPLEMENTARY METHOD DETAILS

A.1 TRANSITION CONTRACTS AND VERIFIER SCORES

The action history is $h_t = (a_1, \ldots, a_{t-1})$. Action types include click, long press, swipe, type, key or system button, open, wait, and terminate. Arguments are type-specific and spatial coordinates are normalized. A trajectory is $\tau = (g, o_1, a_1, \ldots, o_T, a_T, o_{T+1})$.

The deterministic difference $d_t = \mathcal{D}(o_t, o_{t+1})$ tracks page or activity identity, text, focus, input values, task variables, and blocking conditions, including the absence of observable change. A contract contains up to three checkable evidence queries E_t. Its optional recovery specification is

$$\rho_t = (\phi_t, a_t^{\text{rec}}, \hat{d}_t^{\text{rec}}, E_t^{\text{rec}}), \tag{13}$$

where ϕ_t is the failure trigger, a_t^{rec} the corrective action, $\hat{d}_t^{\text{rec}}$ the expected change, and E_t^{rec} the supporting evidence. Validation indicators χ_t assess structural validity, coordinate bounds, admissible action types, and consistency with the proposed action.

The Verifier returns quality scores, a verdict $\nu_t \in \{\text{accept}, \text{partial}, \text{reject}\}$, and confidence $\kappa_t \in \{0, 1, 2, 3, 4\}$. Its score vector is

$$\mathbf{v}_t = (v_t^{\text{fmt}}, v_t^{\text{act}}, v_t^{\text{state}}, v_t^{\text{delta}}, v_t^{\text{progress}}, v_t^{\text{recovery}}, v_t^{\text{safe}}, v_t^{\text{eff}}) \in \{0, 1, 2, 3, 4\}^8. \tag{14}$$

The dimensions assess each candidate's format validity, action validity, state grounding, delta evidence, task progress, recovery quality, loop safety, and cost efficiency. Higher scores indicate stronger support. Recovery receives zero when no failure trigger is active before the candidate action. Otherwise, it assesses whether the action restores required preconditions while preserving completed task state. Loop safety v_t^{safe} supplies the loop-avoidance score in r_{pl}. It assesses avoidance of evidenced ineffective cycles while accounting for changed preconditions and justified retries, backtracking, or waiting. Repetition alone does not establish a loop.

Five verifier dimensions enter the process rewards. Direct action matching supplies r_{type} and r_{arg}, while execution evidence attributable to the evaluated candidate supplies r_{env}. Shared context does not imply shared scores across candidate actions.

A.2 Annotation representation and procedure

The unified transition representation is

$$x_t = (g, o_t, h_t, a_t, o_{t+1}, y_t^{\text{term}}, m_t^{\text{src}}), \tag{15}$$

where y_t^{term} is the available terminal label and m_t^{src} identifies provenance. Next observations and transition differences are available only for samples with observed transitions. Static grounding samples provide localization evidence.

Claude Opus 4.8 annotates transitions using the deterministic state estimate $\bar{s}_t$ and observed difference d_t. Contract acceptance requires semantic action equivalence and valid structure, evidence, and recovery specifications. For each accepted contract, rule-based perturbations of the target or argument generate up to four syntactically plausible hard negatives. The annotator specifies the likely failure, an observable trigger, and a minimal recovery branch. Observed and counterfactual failures retain separate provenance. Preference pairs share the same decision input, and calibration targets cover all eight verifier dimensions.

A.3 Trajectory auditing, hints, and selection

The trajectory Critic uses a fixed Qwen3.5-9B model without a task-specific adapter. It first describes each action and its adjacent screenshot evidence, then assesses trajectory success and every step's reasonableness using these descriptions, execution records, and up to the last three screenshots. After an attempt containing failures or rejected steps, it generates hints restricted to observed failures and behavior to avoid. Correct next actions are excluded from the hints.

An attempt generates hints when the task fails, any step is judged unreasonable, or any step is rejected by the execution or verifier checks. Hints accumulate across attempts as $\eta_i^{(<k)}$. The first attempt has empty hints. Each offline sample preserves the hints available when its reference action was collected. Deployment sets this input to $\emptyset$.

For the retention criterion, z^E requires a successful execution response with no execution error or action invalidity. A verifier rejection vetoes a step only when confidence is at least 3. An invalid verifier assessment gives $b^V = 0$.

We compute task success from all $K = 4$ attempts and retain tasks satisfying

$$\frac{1}{K}\sum_{k=1}^{K} \hat{Y}_i^{(k)} < 1. \tag{16}$$

For attempt length $T_i^{(k)}$, the retained-step fraction and selected attempt are

$$q_i^{(k)} = \frac{\sum_t z_{i,t}^{(k)}}{\max(1, T_i^{(k)})}, \qquad k_i^* = \arg\operatorname*{lex\,max}_k \big(\hat{Y}_i^{(k)}, q_i^{(k)}, -T_i^{(k)}\big). \tag{17}$$

Complete ties favor the earlier attempt. The policy input for each retained action is

$$x_{i,t}^{\text{pol}} = (g_i, I_{i,t}, h_{i,t}, \eta_i^{(<k_i^*)}). \tag{18}$$

Audit evidence $m_{i,t}$ contains the contract, verifier scores, post-action observation, execution outcome, and Critic judgment. These AndroidWorld rollouts supply reinforcement-learning experience separately from the annotation sources.

A.4 Action similarity and environment evidence

The type-specific similarity S_{α^*} uses distance-decayed point similarity for clicks and long presses, direction matching for swipes, token-set F1 for typing, exact or containment matching for application names, and type-specific argument matching otherwise.

The action rewards are $r_{\text{type}} = \mathbf{1}[\hat{\alpha} = \alpha^*]$ and $r_{\text{arg}} = r_{\text{type}} S_{\alpha^*}(\hat{\psi}, \psi^*)$. Invalid completions receive zero total reward. For valid candidates, reference similarity affects only these two rewards.

Candidate-specific evidence. Audit evidence m supplies the current decision context, transition constraints, failure triggers, and available execution records. The Contract Policy binds each candidate to its preconditions, expected change, and recovery specification. The Verifier checks these claims against the task goal and grounded evidence. Generated expectations remain hypotheses; agreement within a generated contract cannot establish transition correctness. All scores below are computed separately for each candidate.

State Delta. The score v^{state} assesses support for the candidate's target, arguments, and preconditions in the current interface. The score v^{delta} assesses evidence for its task-relevant expected transition. Their normalized sum is $r_{\text{sd}} = (v^{\text{state}} + v^{\text{delta}})/8$. Nearby clicks can have similar argument scores yet activate controls with different functions, which this assessment distinguishes when supported by evidence.

Recovery. The reward $r_{\text{rec}} = v^{\text{recovery}}/4$ evaluates the corrective action selected at the current decision point. Credit requires an established failure trigger, restoration of a required precondition, and preservation of completed task state. The score is zero when no trigger is active before the candidate action. An action cannot earn recovery credit for resolving an obstacle that it creates itself.

Progress and loop avoidance. The score v^{progress} assesses advancement toward unfinished task requirements. The score v^{safe} assesses avoidance of previously evidenced ineffective action or state cycles; higher values indicate safer behavior. Assessment accounts for changed preconditions, permitting justified retries, necessary backtracking, and waiting. Repetition alone is insufficient evidence of a loop. The normalized reward is $r_{\text{pl}} = (v^{\text{progress}} + v^{\text{safe}})/8$.

Environment. The binary indicator z_{exec} records successful execution of the evaluated candidate. The binary indicator z_{change} records satisfaction of its task-relevant observable outcome criterion, giving $r_{\text{env}} = (z_{\text{exec}} + z_{\text{change}})/2$. Criteria are action-specific: unrelated visual changes receive no outcome credit, and valid waiting or termination need not produce a screen change. Both indicators require execution records attributable to that candidate under the same decision state. An observed successor state supports only the action that produced it; reference execution success and trajectory success cannot be transferred to unexecuted alternatives.

The verifier-derived scores are normalized to $[0, 1]$, while r_{env} is the average of two binary indicators. Thus, all four process rewards lie in $[0, 1]$. The four process rewards have no action-similarity multiplier. They can vary across candidates even when direct action-matching scores coincide, although the component scores may remain correlated.

A.5 Group-relative optimization

The actor and fixed reference are initialized from the Action SFT policy:

$$\pi_{\theta^{(0)}} = \pi_{\text{ref}} = \pi_{\theta_A^{\text{SFT}}}. \tag{19}$$

For each fixed input x, the sampling policy generates $G = 5$ completions. Parsing precedes scoring, and invalid completions receive zero total reward. For each valid candidate, the Contract Policy constructs a candidate-bound contract and the Verifier assesses it against the shared context and candidate-attributable evidence. These auxiliary assessments occur during candidate execution collection, before the offline policy update. The resulting rewards are $R_j = R(\hat{a}_j, a^*, m)$, with candidate dependence carried by all six components. Their group statistics and advantages are

$$\mu_R = \frac{1}{G}\sum_j R_j, \qquad \sigma_R^2 = \frac{1}{G}\sum_j (R_j - \mu_R)^2, \qquad A_j = \frac{R_j - \mu_R}{\sigma_R + 10^{-4}}. \tag{20}$$

For completion length L_j and token index ℓ, the probability ratio is

$$\xi_{j,\ell} = \frac{\pi_\theta(\hat{a}_{j,\ell} \mid x, \hat{a}_{j,<\ell})}{\pi_{\theta_{\text{old}}}(\hat{a}_{j,\ell} \mid x, \hat{a}_{j,<\ell})}. \tag{21}$$

The sampling policy supplies the ratio denominator, and the fixed SFT reference supplies KL regularization at the same token context. We use clipping radius $\epsilon = 0.2$ and KL coefficient $\beta_{\text{KL}} = 0.01$.

Candidate execution collection alternates with offline policy optimization. At the start of each round, candidates are sampled for each fixed decision input. Each candidate is executed separately after restoring the same decision-state snapshot. The collection stage records its successor observation, execution result, and verifier assessment, preserving their association with that candidate and state.

The candidate set and its evidence are then frozen for GRPO updates. Optimization uses only candidates with complete execution records; missing evidence is neither replaced with zeros nor masked. Newly sampled candidates enter optimization only after execution collection in the next round. Thus, auxiliary scoring and environment interaction occur during collection, while offline updates reuse the resulting candidate-specific records. Predicted outcomes cannot substitute for observed execution evidence.

B Experimental Configuration and Implementation Details

B.1 Implementation and training configuration

We utilize the post-trained Qwen3.5-9B multimodal model (Qwen Team, 2026) as the common base. Action, Contract, and Verifier SFT deploy rank-16 LoRA for two epochs guided by respective learning rates of 2×10^{-5}, 10^{-6}, and 10^{-6}. GRPO leverages rank 64, four epochs, a learning rate of 10^{-6}, $G = 5$, a KL coefficient of 0.01, and a clipping radius of 0.2. Formal distributed training runs across seven 8-GPU nodes equipped with EFA; evaluation scales across 15 8-GPU nodes utilizing a single tensor-parallel-size-1 vLLM instance per GPU. External checkpoints strictly follow their released prompts, parsers, coordinate conventions, and native direct-action pathways. We enforce full sample reporting, rigidly reject runs exhibiting missing samples or API errors, and retain raw predictions alongside per-task records.

B.2 Reward weights

The reward weights follow the order Action Type, Action Argument, State Delta, Recovery, Progress/Loop, and Online Environment:

$$(w_{\text{type}}, w_{\text{arg}}, w_{\text{sd}}, w_{\text{rec}}, w_{\text{pl}}, w_{\text{env}}) = (0.1, 0.7, 0.05, 0.05, 0.05, 0.05). \tag{22}$$

Each coefficient weights its corresponding normalized reward directly. Their effects on group-relative advantages depend on the candidate score distributions and group normalization.

Trajectory auditing is detailed in Appendix A.3.

B.3 Benchmarks and metrics

We evaluate TRAVEC-9B on four offline and two online benchmarks. AndroidControl contains demonstrations from 833 Android applications with explicit instructions, and we report action accuracy over 9,980 test steps per split (Li et al., 2024a). GUI-Odyssey evaluates cross-application mobile navigation over 83,885 steps spanning random, task, device, and app splits, with step success rate, type match, and grounding rate as the evaluation metrics (Lu et al., 2025a). ScreenSpot contains 1,272 grounding queries across mobile, desktop, and web interfaces (Cheng et al., 2024), while ScreenSpot-Pro contains 1,581 queries from high-resolution professional applications (Li et al., 2025a). We report target-hit accuracy on both benchmarks. AndroidWorld provides 116 reproducible task templates across 20 Android applications (Rawles et al., 2025). We evaluate three independently instantiated episodes per template and report episode success rate. MobileWorld evaluates long-horizon cross-application workflows with reproducible backend verification (Kong et al., 2026). Under a GUI-only interaction setting, we evaluate 117 tasks with one attempt per task and report pass-at-one success rate.

B.4 Comparison models

Qwen3.5-9B is a natively multimodal 9B foundation model with a hybrid linear and full-attention backbone and serves as our base model (Qwen Team, 2026). We compare TRAVEC-9B against seven publicly available GUI agents ranging from 7B to 8B parameters. Aguvis-7B-720P is a cross-platform pure-vision agent trained with a two-stage grounding and planning pipeline (Xu et al.,

2024). GUI-R1-7B applies GRPO with unified action-rule rewards on a curated dataset (Luo et al., 2025). ScaleCUA-7B is trained on large-scale cross-platform computer-use data synthesized across six operating systems (Liu et al., 2026d). GUI-Libra-8B combines action-aware supervision with KL-regularized, partially verifiable reinforcement learning (Yang et al., 2026). OpenMobile-8B uses mobile tasks and trajectories synthesized through targeted exploration and policy switching (Cheng et al., 2026). MemGUI-8B-SFT incorporates long-horizon interaction histories through a Context-as-Action framework (Liu et al., 2026a). ForgeQwen3-8B uses MobileForge to adapt the Qwen3-VL checkpoint with automatically generated target-app tasks and hierarchical step-wise feedback (Liu et al., 2026b).

C Case Studies on Transition, Drift, and Recovery

Figure 6 compares model predictions with ground-truth expert trajectories in three representative mobile GUI interaction examples.

C.1 Silent transition failure

Panel (a) shows a scenario where the agent must save a specific location to a shared category. The model predicts a tap targeting the map display controls, while the expert trajectory first navigates back and then swipes left to expose the Save interface. The expert trajectory reaches an intermediate interface state exposing the target Save action, whereas the model prediction selects a different visible control.

C.2 State drift and termination

Panel (b) compares an additional model-predicted interaction with an expert termination after the requested timetable has been displayed. By Step 5, the expert trajectory has reached the timetable state and terminates, whereas the model predicts an additional tap targeting the operator icon. The example highlights a difference in termination behavior after the requested interface state is reached.

C.3 Failure recovery

Panel (c) compares a system-level backward prediction with the expert in-app navigation and dialog-clearing sequence. From the unrelated mathematical exercise, the model predicts a system back command, while the expert selects the native in-app back button, which opens a confirmation dialog. The expert then selects the affirmative option and returns to the primary chapter list.

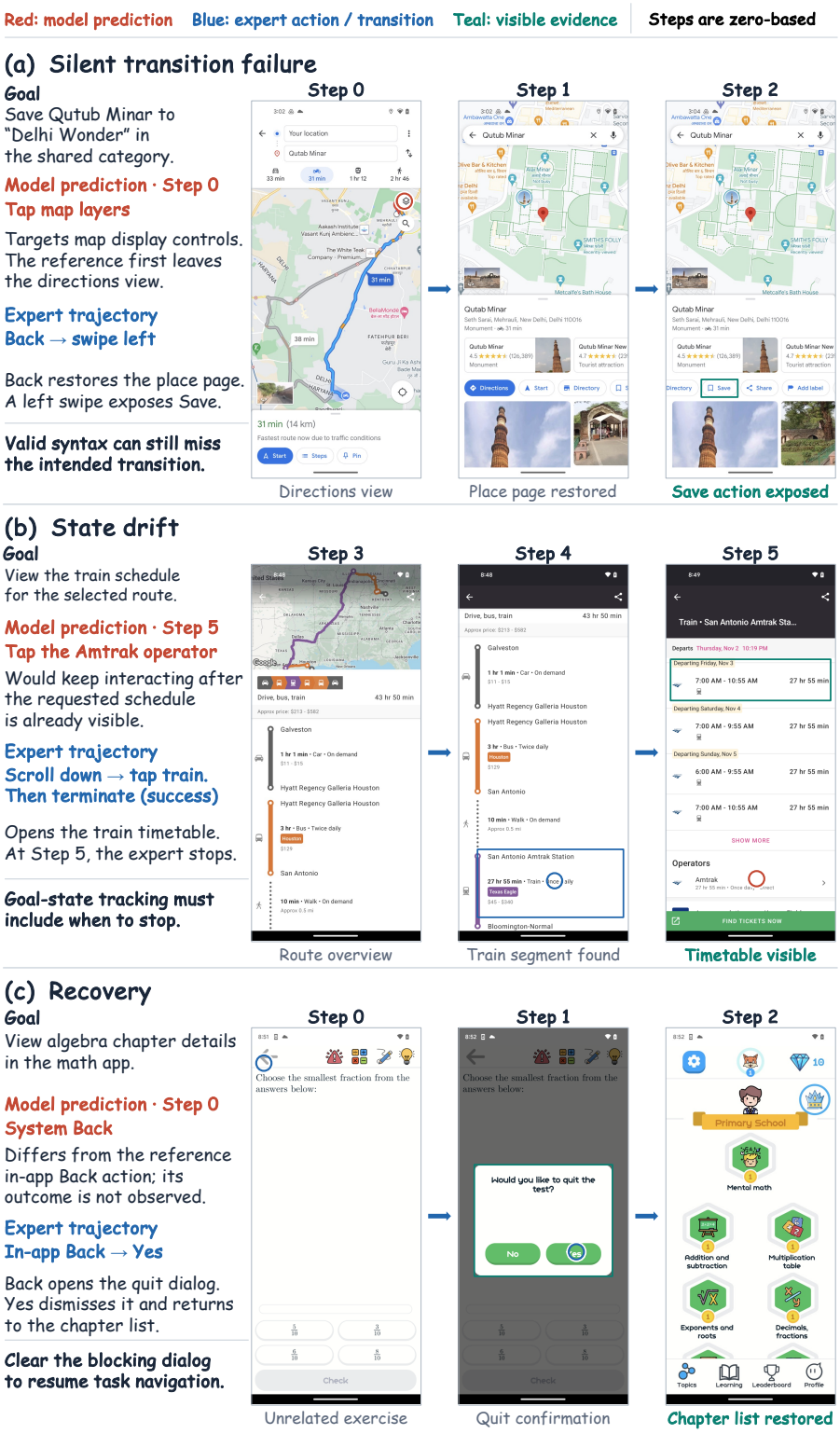


Figure 6: Counterfactual static case studies contrasting unexecuted GUI-agent predictions against ground-truth expert trajectories. Red annotations denote model predictions, blue indicates expert actions, and teal highlights visible evidence.

D ANNOTATED SAMPLES

Figures 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16 present ten illustrative audited contract annotations selected to highlight complementary transition, verification, and recovery patterns. The examples span AndroidControl, AITW, and AMEX and cover seven action types: `tap`, `type`, `wait`, `long_press`, `swipe`, `back`, and `home`.

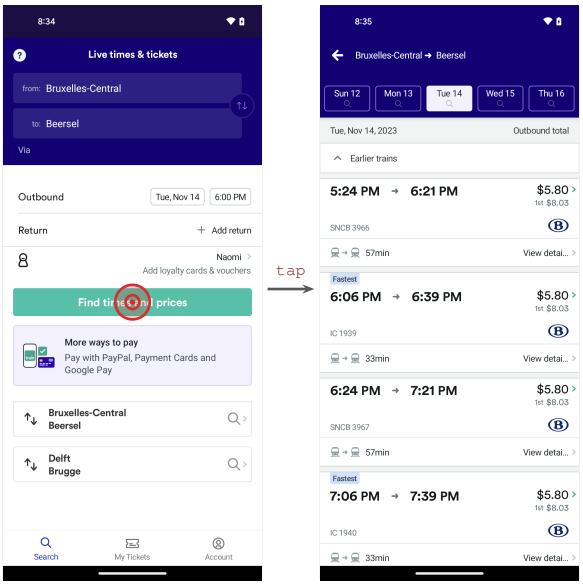
Each example shows the observed state s_t, the annotated action, the resulting state s_{t+1}, and a condensed transition contract. The contract summarizes the grounded task state, action precondition, semantic action, expected state delta, observable appear/disappear evidence ($+$ / $-$), recovery branches, and verifier hints. Action targets are visualized directly on the interface using a red marker for `tap`/`long_press`, a red arrow for `swipe`, and a red box for `type`.

The selected examples illustrate several behaviors targeted by TRAVEC: task progress without page navigation, delayed outcomes requiring temporal verification, repeated or overshooting interactions, recovery from unproductive states, and transitions that prepare later stages of a long-horizon task. These cases show how transition contracts associate action semantics with observable state changes and how recovery annotations specify corrective behavior when the realized transition deviates from the intended one.

Text is condensed for presentation, with truncations marked by "...". Complete annotation records will be released in JSON format upon acceptance.

Goal AndroidControl episode 12896 / step 13 tap

"I have to go to Brussels-Central for work-related reasons. Use the trainline app to locate a rail leaving after 6:00 PM on November 14 from Brussels-Centraal to Beersel."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page`: 'Live times & tickets' (`search_form`) `app`: Trainline `summary`: Trainline journey search form. from: Bruxelles-Central, to: Beersel. Outbound date 'Tue, Nov 14' at '6:00 PM'. A large green primary button labeled 'Find times and prices' is centered on screen, followed by a 'More ways to pay' card ... `key_elements`: 'Find times and prices' primary button (OCR node 18 ...; from: Bruxelles-Central; to: Beersel

PRECONDITION ✓ satisfied conf. 0.95

The 'Find times and prices' button must be visible and enabled, with origin/destination/date populated.

`checks`: 'Find times and prices' button visible near screen center; from field shows 'Bruxelles-Central'; to field shows 'Beersel'

SEMANTIC ACTION

`type`: **`tap`** `point`: `(0.500, 0.516)`

`target`: 'Find times and prices' primary green button

`grounding`: UI_tree_t node id 18, bbox [0.038889,0.491667,0.961111,0.54]

EXPECTED DELTA page_changed = `true`

Tapping the button submits the search and navigates from the journey-search form to the outbound results/timetable page for Bruxelles-Central → Beersel on Tue, Nov 14, 2023.

`+Earlier trains` `+Outbound total` `+Tue, Nov 14, 2023`

`-Find times and prices` `-Live times & tickets`

RECOVERY BRANCHES

- After tap, screen still shows the search form ('Find times ... ⇒ `tap (0.500, 0.516)`
- A blank or loading page appears with no train rows for a ... ⇒ `wait 2s`
- A date picker, time picker, or station editor opened instead ... ⇒ `back`

VERIFIER HINTS progress = `advances` loop risk = `low`

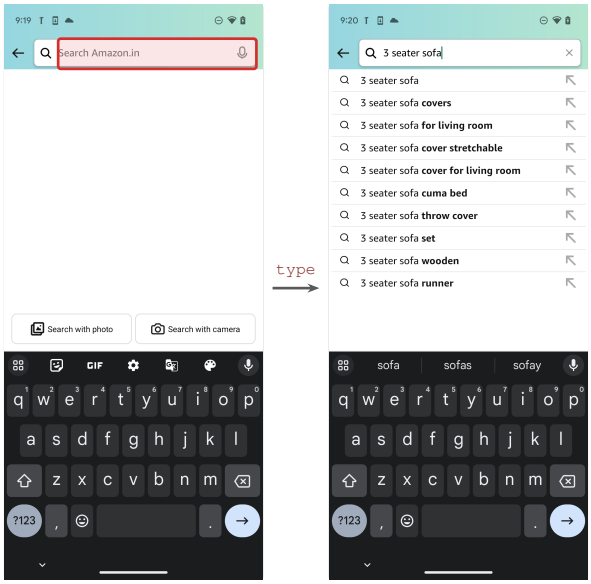
`evidence`: `+text_appeared: Earlier trains` `+text_appeared: Outbound total` `+text_appeared: Tue, Nov 14, 2023`

`expected_page_next`: list_view (train timetable / search results)

Figure 7: (AndroidControl, `tap`). Screenshots and the generated contract annotation (condensed); `item_id`: `bce40caad880af5ccafc5c40`.

Goal AndroidControl episode 14311 / step 2 type

"I want to find a 3 seater sofa set to give to my friend Abdul Karim on his birthday on the 21st of december."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page_type`: `search_screen_with_keyboard` `app`: Amazon `summary`: Amazon in-app search entry screen. The top search input shows placeholder 'Search Amazon.in' and is focused; the soft keyboard is open; 'Search with photo' and 'Search with camera' options are shown above the keyboard. No results or ...

PRECONDITION ✓ satisfied

A focused, editable search field with the soft keyboard already open is the standard precondition for a type action.

`checks`: search field focused; field editable; keyboard visible

SEMANTIC ACTION

`type`: **`type`** `text`: ''3 seater sofa''

`target`: Amazon search input field (placeholder 'Search Amazon.in') at top of screen

`grounding`: `bbox [0.21, 0.06, 0.97, 0.11]`

EXPECTED DELTA page_changed = `true`

The typed text '3 seater sofa' appears in the search field, the placeholder is replaced, a 'Clear search keywords' (X) button appears, and an autocomplete suggestion list renders below the ...

`evidence`: text_appeared: 3 seater sofa; text_appeared: 3 seater sofa covers

RECOVERY BRANCHES

- After typing, the search field still shows placeholder ... ⇒ `tap (0.500, 0.085)`
- The field shows partial or altered text (not exactly '3 ... ⇒ `tap (0.930, 0.085)`
- Keyboard is not visible, so typing cannot proceed ⇒ `tap (0.500, 0.085)`

VERIFIER HINTS loop risk = `low` no_change_risk = `low`

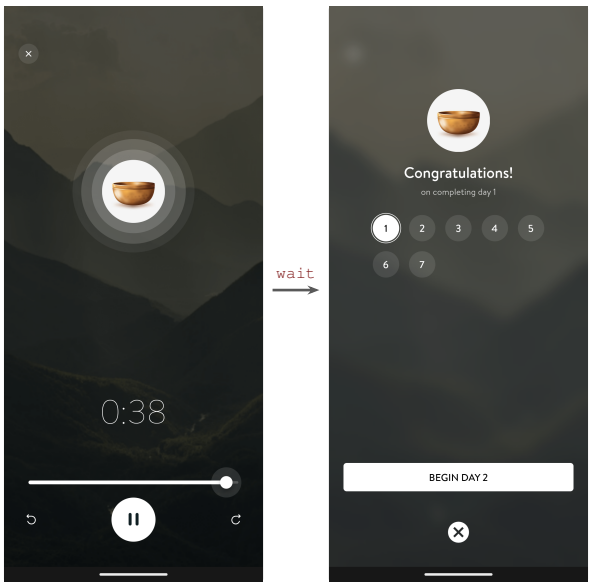
```
evidence:  +text '3 seater sofa' visible in search field   +autocomplete suggestion list contains '3 seater ... +'Clear search keywords' button present -search field still empty/placeholder
```

`notes`: press_enter is false; results/product page has NOT loaded. Verifier must not accept 'search results appeared' as ...

Figure 8: (AndroidControl, `type`). Screenshots and the generated contract annotation (condensed); `item_id`: `2cf51485dd9b973e83c97087`.

Goal AndroidControl episode 12482 / step 2 wait

"Complete the breathing lesson on your own by doing it practically on the Insight Timer app."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE
page_type: media/meditation timer player app: com.spotlightsix.zentimerlite2
summary: Full-screen meditation player over a mountain background with a singing-bowl icon. A timer reads '0:38', a horizontal progress seek bar is near its right end (content_description '94%', focused), and a central round Pause button (——) ...
key_elements: id: 11; role: timer; text: 0:38; id: 12; role: progress_seekbar; text: 94%; id: 13; role: pause_button

PRECONDITION ✓ satisfied
A guided meditation/timer is actively playing (Pause button shown, progress ˜94%); waiting lets the timed session run to completion.
checks: central Pause button visible in screenshot (indicates ...; progress seek bar at '94%' near completion; prior history: tap play then wait, so this wait continues ...

SEMANTIC ACTION
type: **wait**
target: current meditation timer session
intent: Allow the in-progress breathing meditation to run to completion instead of interacting.

EXPECTED DELTA page_changed = true
+Congratulations! +on completing day 1 +BEGIN DAY 2
−0:38 −94%

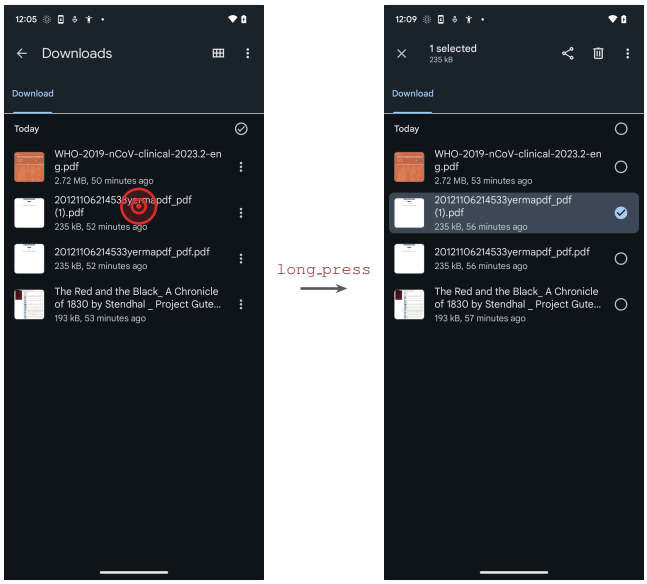
RECOVERY BRANCHES
- After waiting the timer screen is unchanged (still shows ... ⇒ wait
- The player shows a Play (triangle) button instead of Pause ... ⇒ tap (0.500, 0.891)
- Screen changed to something other than the completion screen ... ⇒ back

VERIFIER HINTS loop risk = low no_change_risk = false
evidence: +text 'Congratulations!' present +text 'on completing day 1' present +text 'BEGIN DAY 2' present −timer '0:38'/'94%' still present

Figure 9: (AndroidControl, `wait`). Screenshots and the generated contract annotation (condensed); `item_id`: `a4e5c920a70850d46d3a4ebb`.

Goal AndroidControl episode 20089 / step 2 long_press

"There is a duplicate file of a Yerma pdf that I downloaded; delete the duplicate file from the Google Files app."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page_type:` `list_view` `app:` Files by Google
`summary:` Files by Google 'Downloads' list in list view under a 'Today' section, showing four files: WHO-2019-nCoV-clinical-2023.2-eng.pdf (2.72 MB), two duplicate Yerma PDFs '20121106214533yermapdf_pdf (1).pdf' and ...
`key_elements:` Downloads header with 'Navigate up' back arrow ...; file row '20121106214533yermapdf_pdf (1).pdf, 235 kB' ...; file row '20121106214533yermapdf_pdf.pdf, 235 kB'

PRECONDITION ✓ satisfied

`checks:` Target file row '20121106214533yermapdf_pdf (1).pdf' is ...; expert long_press point [0.518519, 0.35] falls inside that ...; In Files by Google, long-pressing a file row is the ...

SEMANTIC ACTION

`type:` **`long_press`** `point: (0.519, 0.350)`
`target:` duplicate Yerma PDF file row '20121106214533yermapdf_pdf (1).pdf' (235 kB) in the Downloads ...
`intent:` Enter multi-select mode with the duplicate Yerma PDF selected so the delete/Move-to-Trash action becomes available.

EXPECTED DELTA page_changed = `true`

The list enters selection / contextual action mode: a contextual action bar replaces the Downloads header, the long-pressed '(1).pdf' row becomes selected, and delete/share actions appear.

```
+1 selected +235 kB +Cancel
-Downloads -Navigate up
```

RECOVERY BRANCHES

- After the gesture the screen still shows 'Downloads' header ... ⇒ `long_press (0.519, 0.350)`
- '1 selected' appears but the selected row title/subtitle is ... ⇒ `tap 'Cancel (X) button in the ...'`
- A PDF viewer / preview opens instead of selection mode. ⇒ `back 'return to the Downloads list'`

VERIFIER HINTS loop risk = `low`

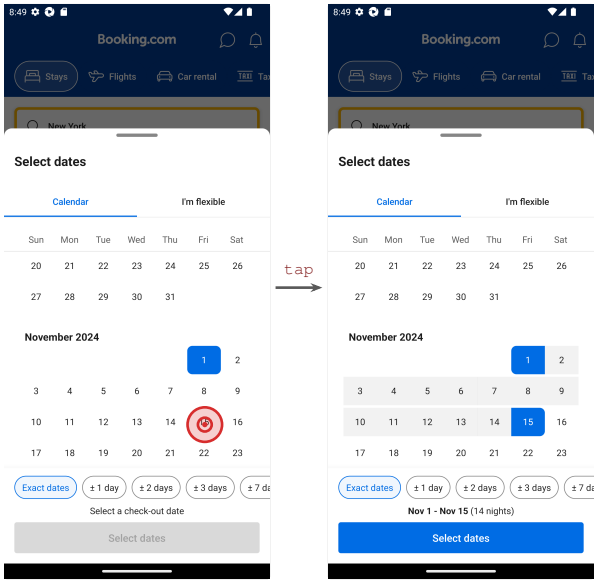
```
evidence:              +OCR/UI-tree contains ' 1 selected'
        +'Move to Trash' action visible
+'20121106214533yermapdf_pdf (1).pdf' node has ...
-'Downloads' title still present
```

`notes:` Do not treat 'possible_popup' as a blocking modal; it is the selection contextual action bar.

Figure 10: (AndroidControl, `long_press`). Screenshots and the generated contract annotation (condensed); `item_id`: `5165d1e151752cac4a1e8c38`.

Goal AMEX episode c7882b1a075f.. / step 9 tap

"Open Booking.com. Search hotels in New York, from 1 November to 15 November. 2 guests and 2 children in 2 rooms with 2 single beds. Filters include balcony. Accept online payments. Sort by distance to downtown."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page_type:` `date_picker_bottom_sheet` `app:` com.booking
`summary:` Booking.com 'Select dates' calendar bottom sheet is open over a dimmed Stays search page. The 'Calendar' tab is active; November 2024 grid is visible. Nov 1 is highlighted blue as the selected check-in date and the prompt 'Select a ...
`key_elements:` 'Select dates' sheet title; Calendar / I'm flexible tabs; November 2024 month grid

PRECONDITION ✓ satisfied

`checks:` date-picker calendar sheet is open; check-in date Nov 1 already selected (blue); 'Select a check-out date' prompt indicates check-out is ...

SEMANTIC ACTION

`type:` **`tap`** `point:` `(0.754, 0.731)`
`target:` November 15 date cell in the calendar grid
`intent:` Tap the day '15' to set the check-out date, completing the Nov 1 - Nov 15 range.

EXPECTED DELTA page_changed = `false`

True
`+Nov 1 - Nov 15 +14 nights`
`-Select a check-out date`

RECOVERY BRANCHES

- After tap, range summary does not show 'Nov 1 - Nov 15' and ... ⇒ `tap (0.754, 0.731)`
- A different day range appears (e.g. Nov 1 - Nov 14 or Nov 1 - ... ⇒ `tap (0.754, 0.731)`
- 'Select dates' button remains greyed after selection ⇒ `tap (0.192, 0.396)`

VERIFIER HINTS progress = `positive` loop risk = `low`

`evidence: +'Nov 1 - Nov 15' text +'14 nights' text +enabled blue 'Select dates' button`
`notes:` Change is a within-sheet content/state update (range completion), not a full navigation; actual_delta labels ...

Figure 11: (AMEX, `tap`). Screenshots and the generated contract annotation (condensed); `item_id`: `0e29249d359784689768be20`.

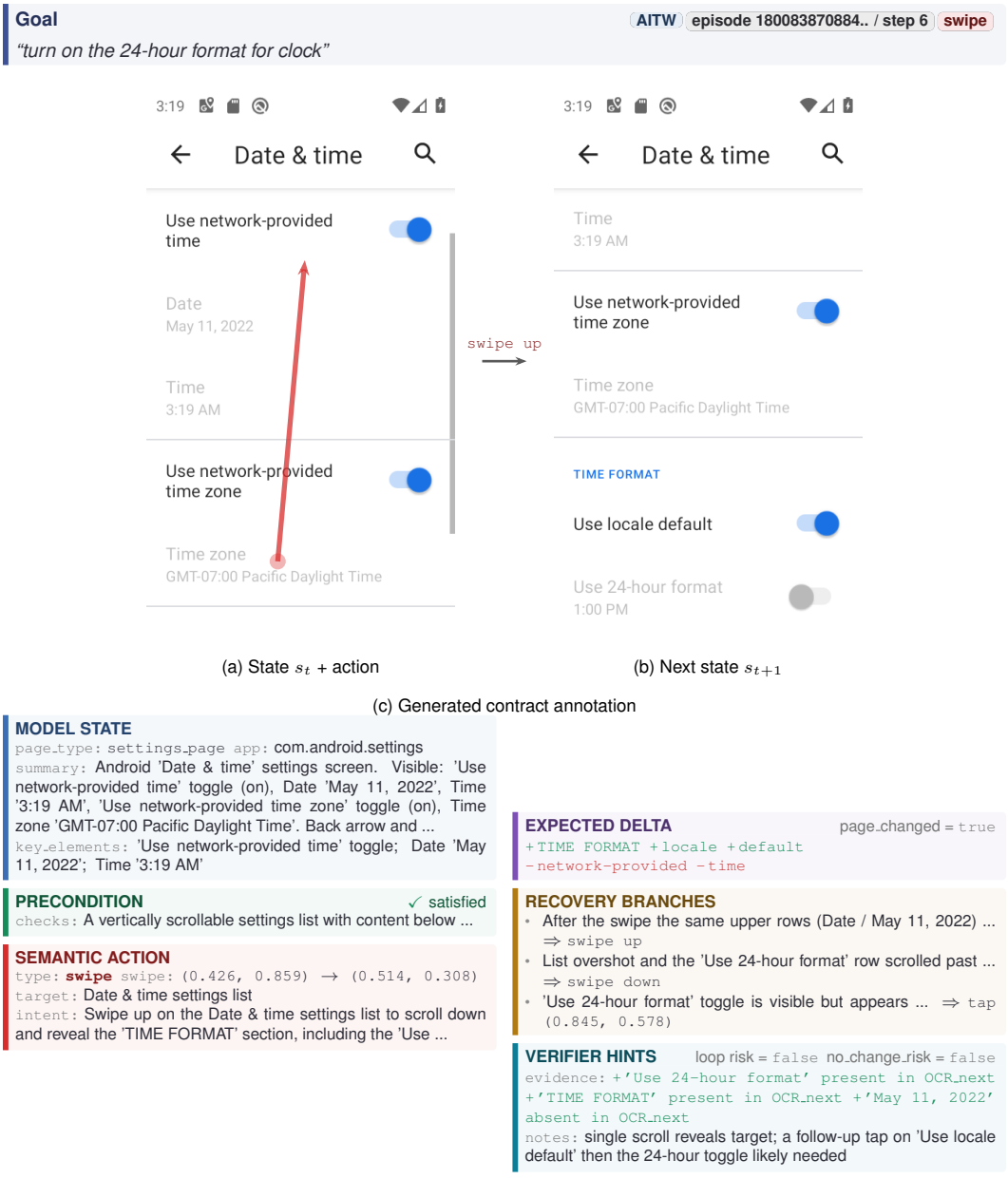
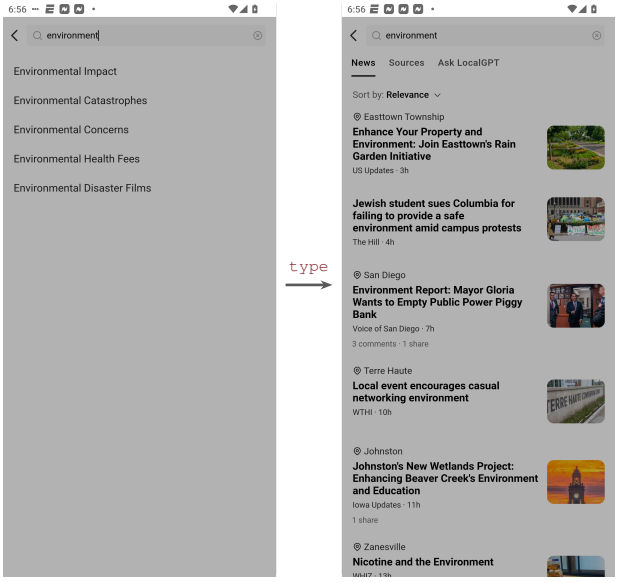


Figure 12: (AITW, `swipe`). Screenshots and the generated contract annotation (condensed); `item_id`: `b94aec170aeb293c4b832cc2`.

Goal AMEX episode 995b4b1d7e8b.. / step 6 type

"Open newsbreak. Search for recent environmental news. Follow the second associated account."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page_type:` `search_screen` `app:` com.particlenews.newsbreak
`summary:` NewsBreak in-app search screen. The search field in the top bar contains the query 'environment' (text cursor after it), with a back arrow at the left and a clear (x) icon at the right. Below the field is an autocomplete suggestion ...

PRECONDITION ✓ satisfied

`checks:` search field shows text 'environment'; autocomplete suggestion list visible below the field; history_actions include type 'environment' into this field

SEMANTIC ACTION

`type:` **`type`**
`target:` search input field in the top bar
`intent:` Submit the already-typed query 'environment' by pressing Enter to run the NewsBreak search.

EXPECTED DELTA page_changed = `true`

`evidence:` results tabs 'News', 'Sources', 'Ask LocalGPT' visible; 'Sort by: Relevance' control visible

RECOVERY BRANCHES

- Autocomplete suggestion list still visible and no results ... ⇒ `type 'search input field'`
- Query text in the search bar changed to a suggestion label ... ⇒ `tap 'clear (x) icon in the ...'`
- Results page appears blank or shows only a loading spinner. ⇒ `wait`

VERIFIER HINTS loop risk = `low` no_change_risk = `low`

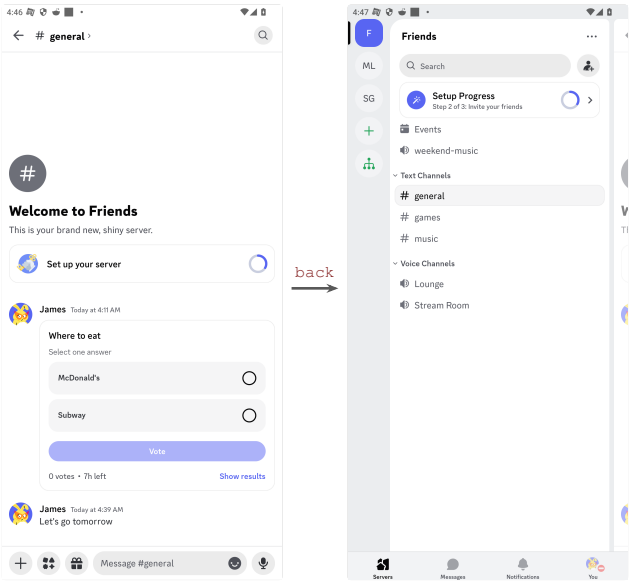
```
evidence:          +'News'/'Sources'/'Ask LocalGPT'
tabs visible       +result list of environment news
items visible      +search bar retains 'environment'
-autocomplete suggestion list still shown
```

`notes:` Advances the search subtask; the goal's 'follow the second associated account' step still requires opening the Sources ...

Figure 13: (AMEX, `type`). Screenshots and the generated contract annotation (condensed); `item_id`: `5103471f383e2e6370f65fe2`.

Goal AMEX episode 7272aff5d48f.. / step 3 back

"Open Discord. Upload and send the most recent two photos to channel "holiday-plans" in server "Study Group"."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page_type:` chat_thread `app:` com.discord
`summary:` Discord viewing the '#general' text channel of the 'Friends' server (top breadcrumb '# general'). Content shows 'Welcome to Friends / This is your brand new, shiny server.', a 'Set up your server' card, a poll from James titled 'Where ...
`key_elements:` breadcrumb '# general' with back arrow at top-left; 'Welcome to Friends' heading; poll 'Where to eat' (McDonald's / Subway / Vote)

PRECONDITION ✓ satisfied

The agent is inside the wrong channel (#general of 'Friends') and must navigate away to reach the correct server/channel. A back press from a channel thread plausibly ...
`checks:` currently inside a channel thread view; back navigation available (top-left arrow / system back); current channel is not the target 'holiday-plans'

SEMANTIC ACTION

`type:` **back**
`target:` system/back navigation out of the '#general' channel thread
`intent:` Exit the current channel view to reach the server and channel list so the correct 'Study Group' server / ...

EXPECTED DELTA page_changed = `true`

The channel thread is replaced by the 'Friends' server navigation drawer: a 'Friends' header with a Search box, a 'Setup Progress' card, 'Text Channels' (general, games, music), 'Voice ...
`evidence:` server/channel drawer visible with 'Friends' header; 'Text Channels' section listing general/games/music

RECOVERY BRANCHES

- After back the same '#general' channel view is still shown ⇒ `back`
- Back exits Discord entirely to the launcher home ⇒ `open_app`
- Navigation drawer is shown but the 'Study Group' (SG) server ... ⇒ `tap 'SG' (Study Group) server ...'`

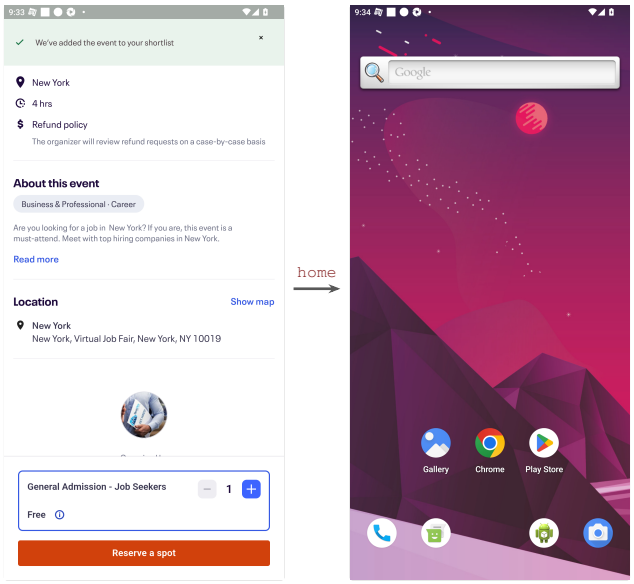
VERIFIER HINTS loop risk = `low` no_change_risk = `low`

`evidence: +'Friends' drawer header visible +'Text Channels' list visible +'SG' server icon visible in server rail -still inside '#general' channel thread`
`notes:` single system back; reversible

Figure 14: (AMEX, `back`). Screenshots and the generated contract annotation (condensed); `item_id`: `6e5f315f466979c0bcef9840`.

Goal AMEX episode 182a9ea23692.. / step 15 home

"Open Eventbrite. Set time to tomorrow. Clear all search filters. Select the third one in New York. Record its location and time in Google Keep Notes. Add it to Favorites."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page_type:` `product_detail app:` com.eventbrite.attendee
`summary:` Eventbrite event detail page for a New York job-fair event. A green success toast at top reads 'We've added the event to your shortlist'. Below are event facts (location 'New York', duration '4 hrs', refund policy), an 'About this ...
`key_elements:` shortlist-added toast; location: New York; duration: 4 hrs

EXPECTED DELTA page_changed = `true`

App foreground changes from the Eventbrite event detail page to the Android launcher home screen.
`+launcher home screen +Google search widget +app icons: Gallery, Chrome, Play Store`
`-Eventbrite event detail content -shortlist toast`

PRECONDITION ✓ satisfied

PRESS_HOME is a global system action that is always available. It is meaningful here because the favorites/shortlist subgoal is already reflected on screen and the next ...
`checks:` Eventbrite event detail page is foreground; system navigation/home affordance is always available; no modal dialog blocks navigation

RECOVERY BRANCHES

- After pressing home the Eventbrite page is still foreground ... ⇒ `home`
- Launcher appears but the agent needs Google Keep, which is ... ⇒ `open_app`
- A different app (e.g., recent app) is shown instead of the ... ⇒ `home`

SEMANTIC ACTION

`type:` **home**
`target:` system home button
`intent:` Leave the Eventbrite event detail page and return to the launcher home screen, a preparatory navigation step toward ...

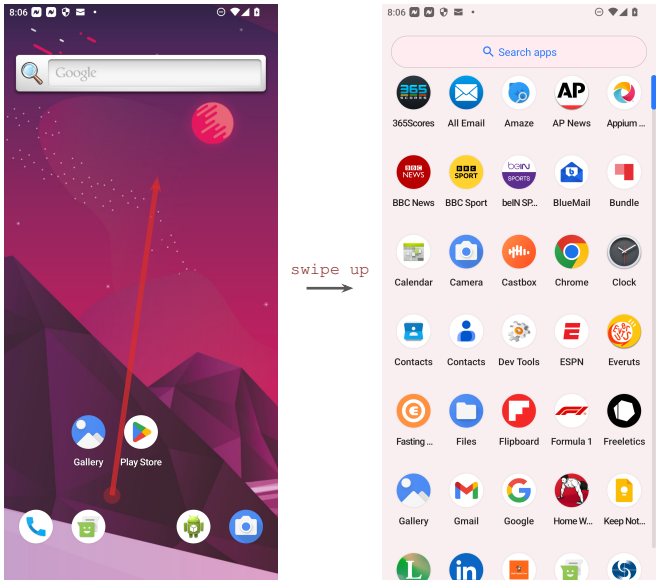
VERIFIER HINTS progress = `advances by ...` loop risk = `low`

`evidence: +screenshot_next is launcher home (not Eventbrite) +page_changed true in actual_delta +image_similarity 0.316 confirms large visual change`

Figure 15: (AMEX, `home`). Screenshots and the generated contract annotation (condensed); `item_id`: `5e5ca51a438767a72724dc96`.

Goal AMEX episode bdf0688f15c0.. / step 1 swipe

"Open Castbox. Explore 'History' category. Play the top podcast, then pause."



(a) State s_t + action (b) Next state s_{t+1}

(c) Generated contract annotation

MODEL STATE

`page_type:` `launcher_home` `app:` com.android.launcher3
`summary:` Launcher home screen: legacy Google search widget near top, two app shortcuts (Gallery, Play Store) mid-lower, and a bottom dock (Phone, Messaging, Android/app, Camera). No app drawer open yet.
`key_elements:` home screen wallpaper; Google search widget; Gallery and Play Store shortcuts

PRECONDITION ✓ satisfied conf. 0.82

On a launcher home screen an upward swipe from the lower area opens the app drawer, which is the standard way to reach apps without a home shortcut (Castbox).
`checks:` home screen visible (wallpaper + widget + dock); no app drawer currently open; swipe originates in lower/central area

SEMANTIC ACTION

`type:` **`swipe`** `swipe:` `(0.394, 0.853)` → `(0.560, 0.298)` `app:` `com.android.launcher3`
`intent:` open the app drawer to locate the Castbox app

EXPECTED DELTA page_changed = `true`

Home wallpaper is replaced by the app drawer overlay: a 'Search apps' bar at top and an alphabetical app grid. Castbox app icon is visible among the listed apps.
`evidence:` 'Search apps' bar appears at top; alphabetical app grid appears

RECOVERY BRANCHES

- Still on home screen; app drawer did not open ⇒ `swipe up`
- App drawer opened but Castbox not visible in current view ⇒ `tap (0.500, 0.083)`
- Wrong page appeared (e.g. recent apps or notifications shade) ⇒ `back`

VERIFIER HINTS loop risk = `low` no_change_risk = `false`

`evidence:` `+actual_delta.page_changed=true` `+image_similarity 0.441` `+Castbox icon visible in screenshot_next drawer`
`notes:` AMEX sample: no OCR/UI tree; delta grounded on screenshot comparison and actual_delta.page_changed. Castbox icon ...

Figure 16: (AMEX, `swipe`). Screenshots and the generated contract annotation (condensed); `item_id`: `974a3a3019b1f3da39adcdac`.