ELEMENT-CONDITIONED TARGET REALIZATION WITH FROZEN GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI execution couples two decisions that need not be learned by the same model: which element the instruction refers to and where to act on that element. We introduce an element-conditioned composition method that assigns target selection to an element selector while preserving the frozen receiver's coordinate generation. Our element selector, FPIR, selects a candidate element and supplies its recorded text, role, and bounds as a hint. On 120 held-out states across four Android applications, FPIR element hints improve frozen receivers' coordinate realization, raising native checker success from 90/120 to 98/120 for MAI-UI-8B, 78/120 to 98/120 for UI-Venus-1.5-8B, and 36/120 to 85/120 for AgentCPM-GUI-8B. A separate supervision audit shows that assigning conflicting action labels to a shared element score distorts the element-learning target. A same-state ablation finds that text and role without bounds retain most of the native execution gains, while the value of spatial bounds depends on the receiver.

1 INTRODUCTION

GUI agents must match a user's target description to the correct interface element. For example, a request for the final museum reservation note must be matched to the correct entry among drafts and other reservation records. Identifying that entry requires matching the user's description to the attributes of the visible candidates.

GUI models such as UI-TARS and MAI-UI learn to ground instructions in screenshots (Qin et al., 2025; Zhou et al., 2025). Specialized grounders such as Aria-UI and GUI-Actor focus on identifying the target screen region (Yang et al., 2025; Wu et al., 2025a). We introduce element conditioning to connect a dedicated target selector with frozen GUI models, and examine how semantic and spatial target information affect downstream performance.

Our selector, Factor-Preserving Intent Realization (FPIR), scores how candidate attributes and interface roles match a supplied step intent. Given the visible candidates, FPIR selects an element and passes its text, role, and bounds to the frozen receiver as a hint. The receiver generates the execution coordinate, and its parameters remain unchanged throughout composition and evaluation.

Our main experiments test FPIR conditioning on 120 held-out click and long-press states across four Android applications. Checker success rises from 90/120 to 98/120 for MAI, 78/120 to 98/120 for UI-Venus, and 36/120 to 85/120 for AgentCPM-GUI-8B. A same-state ablation shows that the selected element's text and role improve all three receivers even without bounds. Bounds alone are less consistent, so the native gains are not simply due to supplying a location. In a separate 48-case study, learning status matching improves element selection and downstream execution with two frozen receivers.

As a supporting analysis, we examine supervision: an action can refer to the correct element but use the wrong interaction mechanism. In this audit, element BCE raises macro element Top-1 from 0.600 to 1.000, and a mechanism-interaction action head also reaches 1.000. This identifies the problem as forcing conflicting action labels onto a shared element score, rather than action-level supervision itself.

Our contributions are:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

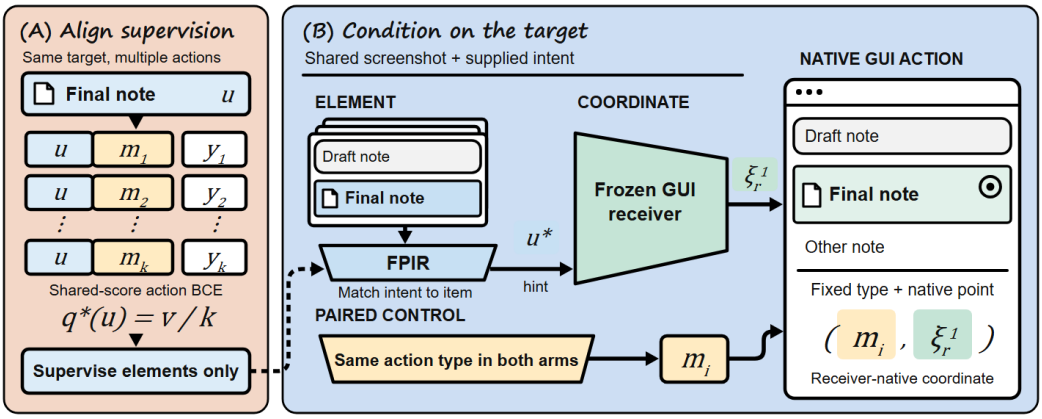


Figure 1: **Element-aligned supervision and element-conditioned composition.** (A) Repeating one element probability across k action BCE labels, v positive, gives $q^*(u) = v/k$; element-aligned supervision instead uses $Y_E(u) = \max_m Y(u, m)$. (B) FPIR selects the target among similar entries in an illustrative interface. Its text, role, and bounds condition a frozen receiver, which generates the native coordinate ξ_r^1. The dashed arrow denotes design motivation.

1. We introduce FPIR-based element conditioning: the selector matches requested and excluded attributes to visible candidates, then supplies the selected element's text, role, and bounds to a frozen receiver. This allows target selection to be improved independently, without retraining the receiver or replacing its coordinate generation.
2. We show that the same selector improves target realization across three frozen GUI receivers. Text and role alone retain most of the native execution gains, while the value of bounds depends on the receiver. Target hints therefore offer more than location guidance, and adding geometric detail is not uniformly beneficial.
3. We show that conflicting action labels distort a shared element score's learning target. In a controlled audit, both element-level supervision and a mechanism-interaction head resolve the resulting element-ranking errors. This supports aligning supervision with what a score is intended to predict, rather than discarding action-level supervision altogether.

2 Related Work

End-to-end GUI action models. ShowUI unifies visual observations, language, and action histories in one vision–language–action model, while UI-TARS learns a unified cross-platform action space and native GUI interaction (Lin et al., 2025; Qin et al., 2025). MAI-UI and AgentCPM-GUI-8B likewise couple GUI grounding with mobile action generation (Zhou et al., 2025; Zhang et al., 2025). These models demonstrate the value of learning a common executable interface. They also motivate our setting: a capable frozen receiver may already provide useful coordinate behavior even when it selects the wrong target.

Action decomposition and stage-wise diagnosis. Several agents already separate high-level reasoning from low-level execution. PGGA conditions low-level actions on procedural plans and exposes a substantial gap between operation, element, and task accuracy (Hsiung et al., 2026). ReGA transfers coordinate prediction from the language model to a location decoder, while GUIPivot uses query inference to bridge coordinate-oriented grounding and action-oriented reasoning (Ni et al., 2025; Wu et al., 2025b). Most directly, CoCo-Agent's conditional action prediction first predicts an action type and then predicts its target conditioned on that type (Ma et al., 2024). FineState-Bench makes a parallel evaluation distinction among localization, interaction, and exact target-state success (Ji et al., 2026). This literature establishes that GUI actions and their failures have stages.

Specialized grounding and whole-action verification. Aria-UI specializes a multimodal model for instruction-conditioned target grounding (Yang et al., 2025). GUI-Actor proposes visual action

regions with an attention head and selects among them with a grounding verifier; notably, its small action head can be trained while the vision-language backbone remains frozen (Wu et al., 2025a). RelationAgent goes beyond isolated widgets by supervising spatial, functional, and hierarchical relations among GUI elements (Ma et al., 2026). In the complementary verification direction, V-Droid learns pairwise process preferences over discretized mobile actions, and IntentScore reranks sampled computer-use actions with an intent-conditioned reward model (Dai et al., 2025; Chen et al., 2026). Grounders provide target-specific evidence but stop before full action realization. Whole-action verifiers score complete candidate actions to improve action selection. Our method intervenes inside action generation: an element selector conditions a frozen receiver that still produces the executable coordinate. We evaluate this interface through paired native execution, separating the contributions of semantic and geometric hint information.

3 Problem Setup

Given a GUI state, a supplied step-level instruction, and visible candidate elements, the goal is to execute an action that satisfies the instruction. Let S denote the state and $Z = (o, t)$ the supplied intent, where o is an operation and t describes the target entity. We study action realization with these inputs, rather than inferring the next intent or discovering candidates.

For the supervision study, a shared proposer exposes element–mechanism candidates $\mathcal{C}(S) = \{a_i = (u_i, m_i)\}_{i=1}^{N}$, where u_i is an observable UI element and $m_i \in \mathcal{M}(u_i)$ is an available mechanism such as click or long press. Multiple candidates may share the same element. Let $Y(u, m) = 1$ when an element–mechanism pair is labeled valid for the supplied intent.

In native execution, a frozen base agent emits an action with a screen coordinate $\hat{\xi}$. For analysis, we represent an action as $a = (u, m, \xi)$, where u is the intended target and m is the action type. The native interface executes (m, ξ), not an element identifier. Selecting u therefore does not guarantee that ξ falls within its responsive region. We ask whether conditioning the frozen receiver on a selected element improves its native coordinate ξ.

Candidate validity and execution success are distinct. A programmatic checker measures success from the resulting state after executing (m, ξ), with an independent reset for each arm. Factor-level diagnostics separately assess candidate coverage, target selection, and coordinate placement.

4 Method

Our method connects FPIR's target selection to a frozen receiver's coordinate generation through an element hint. We first analyze what an element score should predict, then describe the audit scorer, native selector, and hint intervention. Figure 1 summarizes this design.

4.1 Factorize actions and align supervision

Collapse candidates that share one observed element and supervise whether that element admits any valid mechanism:

$$\mathcal{U} = \{u : (u, m) \in \mathcal{C}\}, \qquad Y_E(u) = \max_{m \in \mathcal{M}(u)} Y(u, m). \tag{1}$$

If one shared element probability q is repeated across k action labels, of which v are positive, its local BCE is

$$\mathcal{L}(q) = -v \log q - (k - v) \log(1 - q), \qquad q^* = \frac{v}{k}. \tag{2}$$

For $0 < v < k$, the derivative is $(kq - v)/[q(1 - q)]$, with its unique zero at v/k; boundary cases follow by continuity. This local, unregularized result describes the score target, not an accuracy bound or a guaranteed ranking change. Repeated action labels therefore pull the shared score toward the fraction of valid actions rather than whether any action on the element is valid. In the supervision audit, we train an element score $Q_E(u)$ against $Y_E(u)$ and select $u^* = \arg\max_u Q_E(u)$. Evidence about other action fields does not alter this element decision.

4.2 Element scoring in the supervision audit

The audit uses eight element features: semantic similarity c^{sem} and lexical similarity c^{id} to the requested referent, plus six binary affordance and role flags $\rho(u)$. The element-BCE predictor is

$$\begin{aligned} \phi_E(u;t) &= [c^{sem}, c^{id}, \rho(u)], \\ Q_E(u) &= \sigma\Big(w_E^\top \widetilde{\phi}_E(u;t) + b_E\Big). \end{aligned} \tag{3}$$

Here σ is the sigmoid and $\widetilde{\phi}_E$ is standardized using training-fold statistics. Shared action BCE uses the same predictor with action-level labels; the mechanism-aware controls augment these features with mechanism indicators and, for the interaction head, feature–mechanism products. Table 6 defines the eight shared features.

4.3 Element selection for native execution

The native element selector scores title, team, and status fields, whereas the audit scorer uses text similarity and element roles. The two scorers are trained separately and share the factorization principle. Frozen sentence-encoder cosine supplies title and team distributions p_f^0; a learned status head supplies the status distribution p^L_{status}. Each distribution is normalized over the field values visible in the current state. Let $a_f(u)$ be the candidate's value for field f, z_f the requested title or team, and z^+_{status} and z^-_{status} the requested and excluded status phrases. The score is

$$\begin{aligned} s_{\text{card}}(u) = \sum_{f\in\{\text{title,team}\}} & \log p_f^0(a_f(u) \mid z_f) + \log p^L_{\text{status}}(a_{\text{status}}(u) \mid z^+_{\text{status}}) \\ & + \log[1 - p^L_{\text{status}}(a_{\text{status}}(u) \mid z^-_{\text{status}})]. \end{aligned} \tag{4}$$

The score rewards matches to the requested title, team, and status, and penalizes matches to the excluded status. We replace status alone: learning all three relations repairs status but adds title errors in development. This selector is fixed before evaluation.

4.4 Learning and native action composition

All learned components are small regularized heads over frozen features. In the supervision study, the element head uses element-level BCE, while the whole-action comparators use action-level BCE. The status head is trained on independently generated, canonically labeled paraphrases. Encoders and GUI agents remain unchanged, and no native outcome updates any relation.

For native generation, the frozen receiver decodes and parses one action from the screenshot and intent, either without a hint or with the selected element's text, role, and normalized bounds. The receiver generates both coordinates natively.

Native composition. Given $S, Z, \mathcal{U}$, the native element selector, receiver r, and action type m_i for state i:

1. Select $u^* = \arg\max_{u\in\mathcal{U}} s_{\text{card}}(u)$ and read its recorded text, class role, and bounds to form $h^* = \text{hint}(u^*)$.
2. Decode receiver r without the hint to obtain coordinate ξ_r^0, and with h^* to obtain coordinate ξ_r^1.
3. Execute (m_i, ξ_r^0) and (m_i, ξ_r^1) from independent resets.

The hint provides the selected element's attributes and bounds; the receiver predicts the execution point.

5 Evaluation

The main evaluation asks whether element evidence improves native execution through frozen GUI receivers, using tasks that open or select a target item in four Android applications. A controlled audit examines the supervision unit that motivates the element-level interface.

5.1 EVALUATION STUDIES

Supervision audit. We isolate label dilution on 15 application-held-out AndroidWorld states with 344 action candidates collapsing to 222 elements. The audit changes the training label and head structure while holding test elements and base features fixed.

Native evaluation. The main test evaluates FPIR conditioning on 120 new four-item states: 30 Broccoli click, 30 Pro Expense click, 30 Markor long press, and 30 Files long press. The ten topics and every visible topic/status/team triple are disjoint from development. Each arm starts from an independent AndroidWorld reset and ends in a programmatic checker (Rawles et al., 2025). Click tasks succeed when the intended item's detail view opens; long-press tasks succeed when the intended item enters the selected state. Frozen AgentCPM-GUI-8B is evaluated with the same states, candidates, FPIR selections, reset procedure, and checkers (Zhang et al., 2025). A separate 48-case, six-card study evaluates the learned status head with two frozen receivers.

5.2 NATIVE INTERVENTION DESIGN

For receiver r and state i, let ξ_{ri}^0 be the coordinate generated without an element hint and ξ_{ri}^1 the coordinate generated after FPIR supplies the selected element. To isolate the effect of target information on coordinate realization, we hold the action type fixed across paired conditions and let the frozen receiver generate the coordinates. For each state i, the shared type m_i is taken from the unconditioned MAI prediction before checker evaluation and reused for every receiver:

$$A_{ri}^0 = (m_i, \xi_{ri}^0), \qquad A_{ri}^F = (m_i, \xi_{ri}^1). \tag{5}$$

Each arm starts from an independent reset. Let Y_{ri}^a be its checker outcome for $a \in \{0, F\}$. For n paired states, the observed element-conditioning effect is

$$\widehat{\Delta}_E^{(r)} = \frac{1}{n} \sum_{i=1}^{n} \left(Y_{ri}^F - Y_{ri}^0 \right). \tag{6}$$

This effect includes the frozen receiver's response to the hint.

5.3 COMPARATORS AND METRICS

Comparators. The supervision audit compares shared action BCE, a mechanism-interaction action head, and element BCE under matched features, regularization, folds, and test elements. The native evaluation compares the unconditioned and FPIR-conditioned actions in Equation 5. On the same 120 states, a hint-information ablation also compares text plus role and bounds alone with the no-hint and full-hint conditions. The selected element, screenshot, intent, reset, and checker are unchanged. The supporting status-head study compares raw signed MiniLM relations with the status-head score in Equation 4.

Frozen generation protocol. FPIR, prompts, candidates, and receiver weights are set before the corresponding generations. MAI and UI-Venus decode each unique request once at temperature zero. AgentCPM-GUI-8B uses its prequalified official configuration (temperature 0.1, top-p 0.3), one response per request, and a predetermined seed. Action conversion details are given in the appendix.

Endpoints and paired analysis. Offline selection uses tie-aware Top-1. Native evaluation reports coverage, strict checker success, paired success-rate differences, and wins/losses/ties (W/L/T). Wins and losses are state-level improvements and regressions relative to the comparison arm; ties include unchanged successes and failures. We report two-sided exact sign-test p-values over discordant pairs. Checker outcomes never enter scoring. The main analysis contains 120 paired states for each of three receivers, or 720 declared arms. Method failures, including parsing failures, count as unsuccessful trials. Only explicit infrastructure failures are eligible for retry. The terminal checker determines success.

Table 1: Supervision audit: held-out macro element and whole-action Top-1.

Training unit	Element	Whole action
Shared action BCE	0.600	0.300
Interaction action BCE	1.000	0.333
Element BCE	1.000	–

Table 2: Paired native checker success with shared action type. Counts are out of 120. The receiver generates both coordinates; FPIR-conditioned adds only the selected element hint.

Receiver	No element hint	FPIR element hint	Gain (pp)	W/L/T	Exact p
MAI-UI-8B	90	**98**	+6.7	9/1/110	0.0215
UI-Venus-1.5-8B	78	**98**	+16.7	21/1/98	1.10×10^{-5}
AgentCPM-GUI-8B	36	**85**	+40.8	54/5/61	1.91×10^{-11}

6 RESULTS

The supervision audit tests label design; the native study tests the element-conditioning interface and the information it conveys to each receiver.

6.1 ELEMENT-LEVEL LABELS AVOID SHARED-SCORE CONFLICTS

Conflicting action labels. The matched comparison changes the supervision unit while holding element features fixed. Shared action BCE reaches 0.600 application-macro element Top-1, whereas element BCE reaches 1.000 (Table 1). Ten correct elements have $k = 2, v = 1$, so repeated action labels impose a local target of $1/2$ rather than one. This is the empirical counterpart of Equation 2.

Mechanism-interaction control. A mechanism-interaction action head also recovers 1.000 element Top-1. The problem is therefore not action-level supervision itself, but forcing conflicting mechanism labels onto one shared element score. Separating mechanisms in the representation also recovers element ranking, whereas element BCE directly supervises whether any mechanism makes the element valid. The interaction head's unaggregated whole-action Top-1 is still 0.333: correct element ranking does not guarantee a correct complete action. Full control and calibration diagnostics are in the appendix.

6.2 ELEMENT CONDITIONING IMPROVES TARGET REALIZATION

Target realization improves for all three receivers. FPIR conditioning improves native checker success for all three evaluated receivers (Table 2). MAI rises from 90 to 98 successes, with 9/1/110 wins/losses/ties ($p = 0.0215$). UI-Venus rises from 78 to 98, with 21/1/98 ($p = 1.10 \times 10^{-5}$). AgentCPM-GUI-8B rises from 36 to 85, with 54/5/61 ($p = 1.91 \times 10^{-11}$). Candidate coverage is complete, and FPIR selects the correct target in 109/120 states.

The gain varies by receiver, from 6.7 to 40.8 percentage points, even though the selector and hint are fixed. Each frozen receiver translates the hint into its own coordinate.

Gains extend across applications. Success increases in 11 of the 12 receiver–application combinations (Table 3); the remaining combination, MAI on Broccoli, stays at 30/30. The aggregate improvement is therefore not confined to one application. Files remains the hardest application after conditioning, with 14/30, 16/30, and 13/30 successes for MAI, UI-Venus, and AgentCPM, respectively. Target evidence helps across applications, but does not remove all realization errors.

Semantic cues carry much of the gain. A same-state ablation separates text and role from numerical bounds while holding the FPIR selection fixed (Table 4). Text and role without bounds retain most of the gain for MAI and UI-Venus and yield the highest success for AgentCPM. Bounds alone help MAI and UI-Venus but hurt AgentCPM. Thus the full-hint gains are not solely an effect of supplying location. Adding bounds to text and role changes success by $+2$, $+5$, and -2 cases,

Table 3: Application-level native checker success. Each cell shows no hint → FPIR hint, out of 30 states.

Application	MAI-UI-8B	UI-Venus-1.5-8B	AgentCPM-GUI-8B
Broccoli	$30 \rightarrow 30$	$29 \rightarrow 30$	$8 \rightarrow 27$
Pro Expense	$20 \rightarrow 24$	$11 \rightarrow 22$	$7 \rightarrow 16$
Markor	$29 \rightarrow 30$	$28 \rightarrow 30$	$15 \rightarrow 29$
Files	$11 \rightarrow 14$	$10 \rightarrow 16$	$6 \rightarrow 13$

respectively; all three paired p-values exceed 0.05. With selection held fixed, this ablation measures how each receiver uses the hint. Detailed paired statistics appear in Appendix B.1.

Table 4: Hint-information ablation: native checker successes out of 120 states per receiver. No hint and full hint reuse the main comparison; text + role and bounds only are additional conditions on the same states.

Receiver	No hint	Text + role	Bounds only	Full hint
MAI-UI-8B	90	96	92	98
UI-Venus-1.5-8B	78	93	88	98
AgentCPM-GUI-8B	36	87	31	85

Effect of the learned status head. In the 48-case study, replacing raw MiniLM status compatibility with the learned status head raises target selection from 24/48 to 40/48. Native success rises from 26/48 to 42/48 for MAI and from 24/48 to 37/48 for UI-Venus (Table 5). Both receivers use the same updated element selections; the development comparison with learning all relations appears in Table 11.

The execution gains are concentrated in the 16 states where the status-head selector corrects the raw selector's target choice. Within this group, MAI success rises from 0 to 16 and UI-Venus from 1 to 14. Success counts are unchanged in the groups where both selectors are correct or both are wrong. This supporting study connects improved target selection to downstream execution gains while keeping the receivers frozen.

Table 5: Native success by selector correctness in the separate 48-case study. Each receiver column shows raw MiniLM relations → learned status head. Group sizes sum to 48; no state is correct only under the raw selector.

Selector correctness	States	MAI-UI-8B	UI-Venus-1.5-8B
Status-head selector only correct	16	$0 \rightarrow 16$	$1 \rightarrow 14$
Both correct	24	$23 \rightarrow 23$	$22 \rightarrow 22$
Both wrong	8	$3 \rightarrow 3$	$1 \rightarrow 1$

The native studies show that element hints can improve execution across frozen receivers, with semantic cues carrying much of the benefit.

7 Discussion

One selector across three receivers. The same selector and hint improve all three frozen receivers, with receiver-dependent gains. Offline grounding accuracy measures whether the selector selects the right object; paired native execution measures how the receiver uses the element hint. The hint ablation shows that semantic element cues account for much of the benefit; numerical bounds have receiver-dependent value.

Element-level supervision. The supervision audit highlights a separate design choice: what an element score is trained to predict. Conflicting labels from complete actions can pull the score away from element correctness. Both element labels and a mechanism-aware representation resolve the

ranking errors in the audit. The distinction is between an element score and an action score, rather than between universally suitable and unsuitable label types.

Implications for system reuse and evaluation. The evaluated composition improves target realization with small selector heads while keeping the GUI receiver frozen, supporting reuse of independently developed selectors and receivers. The paired design connects candidate selection, coordinate generation, and checker outcomes, complementing stage-wise benchmarks and transition verifiers (Ji et al., 2026; Guo et al., 2026). Very recently, Jev (Almeida, 2026) introduced a model for structured probabilistic decisions, with limited public detail on its architecture and training procedure at the time of writing. In GUI target realization, our findings suggest that the value of a specialized decision component depends not only on selecting the right target, but also on the information it conveys and how the downstream receiver uses that information.

8 Limitations

The evaluation establishes results for supplied-intent realization with four candidate items per state, covering click and long-press workloads in four controlled applications. Extending this setting to autonomous planning and open-ended candidate discovery would require end-to-end evaluation. The semantic holdout tests new topics, compositions, instructions, and screens within a fixed vocabulary of status and team classes. Transfer to new attribute classes is a further evaluation direction. FPIR's 109/120 target selections and the remaining responsive-region errors in Files identify opportunities to improve selection and coordinate realization separately. Broader actions and multi-step tasks remain for future native evaluation.

9 Conclusion

FPIR improves target realization by supplying target evidence to frozen GUI receivers, each of which generates its own execution coordinate. In paired native tests, this intervention raises checker success across three receivers. The supervision audit identifies why a shared element score should not absorb conflicting action labels. Text and role alone carry substantial gains, while adding numerical bounds has receiver-dependent effects.

AI Usage Disclosure

We used generative AI tools for polishing the manuscript and assisting with information searching. We also used these tools for feedback on experimental design and assistance with implementation.

References

Diogo Almeida. Introducing system one models & Jev. TypeSafe AI Blog, September 2026. URL `https://typesafe.ai/blog/introducing-system-one-models-and-jev`.

Rongqian Chen, Yu Li, Zeyu Fang, Sizhe Tang, Weidong Cao, and Tian Lan. Intentscore: Intent-conditioned action evaluation for computer-use agents. *arXiv preprint arXiv:2604.05157*, 2026.

Gaole Dai, Shiqi Jiang, Ting Cao, Yuanchun Li, Yuqing Yang, Rui Tan, Mo Li, and Lili Qiu. Advancing mobile gui agents: A verifier-driven approach to practical deployment. *arXiv preprint arXiv:2503.15937*, 2025.

Linqiang Guo, Wei Liu, Li Gu, Yang Wang, et al. Stepreflect: Structured ui transition reflection for mobile gui agents. *arXiv preprint arXiv:2608.05587*, 2026.

Lei Hsiung, Zhiyu Chen, Seonhoon Kim, and Qun Liu. Pgga: A plan-grounded gui agent for automated device support. In *Proceedings of the 4th Workshop on Advances in Language and Vision Research (ALVR)*, pp. 105–114, 2026.

Fengxian Ji, Jingpu Yang, Zirui Song, Yuanxi Wang, Zhexuan Cui, Yuke Li, Qian Jiang, and Xiuying Chen. Finestate-bench: Benchmarking state-conditioned grounding for fine-grained gui state

setting. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 43073–43088, 2026.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19498–19508. IEEE, 2025.

Longhui Ma, Di Zhao, Siwei Wang, Zhao Lv, and Miao Wang. Beyond element-level understanding: Explicit relational understanding for gui agents. *Pattern Recognition*, 176:113262, 2026.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9097–9110, 2024.

Feiyue Ni, Yanchu Guan, Yuchong Sun, Dong Wang, Chenyi Zhuang, Jinjie Gu, and Ruihua Song. Rega: Reasoning and grounding decoupled gui navigation agents. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 375–387. Springer, 2025.

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.

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*, volume 2025, pp. 406–441, 2025.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 15101–15128, 2025a.

Zongru Wu, Pengzhou Cheng, Zheng Wu, Tianjie Ju, Zhuosheng Zhang, and Gongshen Liu. Smoothing grounding and reasoning for mllm-powered gui agents with query-oriented pivot tasks. *arXiv preprint arXiv:2503.00401*, 2025b.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22418–22433, 2025.

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 *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 155–180, 2025.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025.

A Implementation and Reproducibility Details

A.1 Supervision and relation heads

The supervision-audit bank contains 344 action rows that collapse to 222 observable elements in 15 episodes. All four heads use the same eight element features, MiniLM-L6-v2 embeddings, and leave-one-application-out folds. Each fold fits a standardized logistic regression with $C = 1$, at most 1,000 iterations, and seed 17. The action heads use shared features, a two-way mechanism indicator, or the indicator plus all feature–mechanism interactions. Their element scores are the maximum action score over mechanisms. Element BCE instead uses one existential label per element. Table 6 specifies the shared features.

Table 6: Feature definitions for the supervision study and element scorer.

Feature group	Definition
Semantic similarity (1)	Cosine similarity between the requested referent and the element descriptor, which contains visible carrier text, class role, and clickability flags.
Lexical similarity (1)	Maximum normalized edit similarity over the element's carrier strings: $1 - d_{\text{edit}}(s,t)/\max(\lvert s\rvert, \lvert t\rvert)$. Strings are case-folded and reduced to space-separated ASCII alphanumeric tokens; empty strings receive zero similarity.
Affordance and role flags (6)	Clickable, long-clickable, and four class-name indicators: `textview`, `layout`, `image`, and editable (`edit` or `autocomplete`).

The native study uses the fixed selector with a learned status head in Equation 4. Its sentence encoder is `sentence-transformers/all-MiniLM-L6-v2`, with unit-normalized outputs. Each of the title, status, and team fields has 12 classes and 108 canonically labeled training examples, including the canonical names. Each multinomial linear head is initialized from class-name embeddings with zero bias and trained for 300 full-batch Adam steps at learning rate 0.03 and seed 365. The loss is multiclass cross-entropy plus $0.01\ \text{mean}[(W - W_0)^2]$, where the rows of W_0 are the unit-normalized, field-prefixed class-name embeddings. This penalty anchors weights to their initialization; it does not regularize the bias. The test-time selector retains frozen cosine for title and team and uses the learned head only for status. The final epoch is used without a hyperparameter or checkpoint sweep.

A.2 Element scores

Equation 3 defines the supervision-audit predictor. The element adapter used in supporting studies uses semantic similarity c^{sem}, continuous lexical similarity c^{id}, six binary affordance and role flags $\rho(u)$, and editability $e(u)$:

$$\begin{aligned} \phi_F(u;t) &= [c^{sem}, c^{id}, \rho(u), c^{id}(1-e(u))], \\ s_F(u) &= w_F^\top \widetilde{\phi}_F(u;t) + b_F, \quad P_E^F(u) = \text{softmax}_{u\in\mathcal{U}}\, s_F(u). \end{aligned} \tag{7}$$

Here $\widetilde{\phi}_F$ denotes features standardized using training-fold statistics. The flags in $\rho(u)$ are the six binary features listed in Table 6; $e(u)$ is the editable flag. The adapter adds $c^{id}(1-e(u))$ to the eight shared features.

For the native element selector, φ is a fixed, unit-normalized sentence encoder, and $\mathcal{V}_f(S)$ contains the field values visible in S. The distributions used in Equation 4 are

$$\begin{aligned} p_f^0(v \mid z) &= \text{softmax}_{v\in\mathcal{V}_f(S)} \left(\varphi(f{:}z)^\top \varphi(f{:}v)\right), \quad f \in \{\text{title}, \text{team}\}, \\ p_{\text{status}}^L(v \mid z) &= \text{softmax}_{v\in\mathcal{V}_{\text{status}}(S)} \left(W_{\text{status}}\varphi(\text{status}{:}z) + b_{\text{status}}\right)_v . \end{aligned} \tag{8}$$

A.3 Native states, candidates, and prompts

The main bank contains 120 states: 30 per application in Table 7. Each state has four visible item candidates. The ten topics and all visible topic–status–team triples are disjoint from development. Candidate extraction and FPIR scoring occur before checker outcomes are read. Candidate coverage is 120/120.

The unconditioned request contains one screenshot, the supplied intent, and an empty action history. The conditioned request appends the selected element's text, role, and bounds, together with an explicit instruction to use that element without switching targets. The exact conditioning template and per-request prompts are provided in the accompanying reproducibility package. The bounds use independent 0–999 normalization in screenshot coordinates. Each receiver receives the same screenshot and base intent in both arms. No candidate-center coordinate or oracle field enters generation or execution.

A.4 Frozen receivers and action conversion

We evaluate `Tongyi-MAI/MAI-UI-8B`, `inclusionAI/UI-Venus-1.5-8B`, and AgentCPM-GUI-8B. MAI and UI-Venus use greedy decoding with one generation per unique request

Table 7: Tasks and success criteria. Success is evaluated from the post-action accessibility state and depends on the target label, not the executed route or annotated coordinate box.

Application	Action type	Candidate element	Success criterion
Broccoli	click	recipe card	detail view contains target and Navigate up
Pro Expense	click	expense row	detail view contains target and EDIT
Markor	long press	note row	description starts "Selected <target>"
Files	long press	file row	target row has selected state

and maximum lengths of 512 and 1,024 new tokens, respectively. UI-Venus uses its mobile prompt and 0–999 pointer convention. AgentCPM uses its official mobile action schema, temperature 0.1, top-p 0.3, and one predetermined per-request seed. Its `POINT` is converted from the official 0–1000 frame. A missing duration takes the official 200 ms default; duration at most 200 ms maps to click and larger values map to long press.

For each receiver, parsed native coordinates are scaled back to the 1080×2400 Android screen. The unconditioned arm uses the receiver's coordinate without a hint; the FPIR-conditioned arm uses its coordinate after the element hint. Both arms execute from independent resets.

A.5 Metrics, failures, and paired tests

Tie-aware Top-1 assigns an episode the fraction of maximum-scoring tied candidates that are valid. Rank is the one-indexed rank of the highest-scoring valid candidate. Valid mass is the softmax mass assigned to all valid elements. BCE and Brier are element-level averages. Action-head probabilities are first max-aggregated over mechanisms; the element head supplies its probability directly. The aggregated action probabilities are ranking diagnostics rather than calibrated existential probabilities.

Native checker success always uses all declared states as the denominator. Coverage, element, view, and coordinate diagnostics are recorded separately and do not gate execution. Missing, unparsable, unsupported, or coordinate-free actions are method failures and receive checker failure. Infrastructure failures are reported separately and are eligible for the declared retry policy. Oracle-box containment is recorded only after an action executes; a method failure therefore contributes zero to that diagnostic.

For every adjacent arm comparison, wins and losses are state-level checker changes, ties contain both unchanged successes and unchanged failures, and the reported value is a two-sided exact sign test over discordant pairs.

B Complete Results and Secondary Diagnostics

B.1 Native paired statistics and execution accounting

Table 8 gives the paired evidence behind the main native results. Each row contains 120 states. Wins and losses denote checker changes from failure to success and success to failure, respectively; ties include both unchanged successes and unchanged failures.

Table 8: Paired native outcomes for unconditioned versus FPIR-conditioned coordinates. Two-sided exact sign-test p-values use the discordant pairs.

Receiver	Success pair	W/L/T	Exact p
MAI-UI-8B	$90 \rightarrow 98$	9/1/110	0.0215
UI-Venus-1.5-8B	$78 \rightarrow 98$	21/1/98	1.10×10^{-5}
AgentCPM-GUI-8B	$36 \rightarrow 85$	54/5/61	1.91×10^{-11}

Hint information ablation. The main no-hint and full-hint outcomes are paired with two new executions per state: text and role without bounds, and bounds without text or role. The FPIR selection, screenshot, and intent do not change. All 720 new native arms are accounted for. Table 4 reports checker success for the four conditions.

Text and role versus no hint yields 7/1/112 wins/losses/ties for MAI ($p = 0.0703$), 16/1/103 for UI-Venus ($p = 2.75 \times 10^{-4}$), and 54/3/63 for AgentCPM ($p = 4.29 \times 10^{-13}$). Full hint versus text and role gives 2/0/118 ($p = 0.5$), 5/0/115 ($p = 0.0625$), and 5/7/108 ($p = 0.774$), respectively. These paired contrasts show substantial benefit from identity cues without numeric bounds, while the incremental value of bounds varies by receiver. AgentCPM emitted a directional swipe in 14 bounds-only requests versus one text-and-role request; such outputs remain unsuccessful in the one-step click/long-press protocol.

Execution accounting. The main comparison contains 720 declared arms: two per state for each of three receivers. The hint ablation adds 720 new arms in a separate archive. No canonical arm is an infrastructure failure, and method failures remain unsuccessful trials. To ensure clean initial state in the affected executions, we re-executed all 240 Files arms and three AgentCPM Markor arms using the already fixed model predictions. All 243 starts passed clean-state and candidate-geometry validation. Target-presence and exact-target-only checker variants agree on all 243 replayed arms. This procedure corrects execution initialization while preserving the semantic state bank and all model predictions.

Files coordinate diagnostic. In Files, the stored candidate box covers the filename text rather than the whole selectable tile. Box-containment counts therefore do not describe all valid touch locations; the terminal checker evaluates which file is selected.

B.2 Full supervision controls

Table 9 expands the supervision audit with the additive control and calibration diagnostics. A dense-grid stress over $1 \leq v \leq k \leq 8$ numerically recovers v/k within grid resolution while the element target remains one. A mechanism indicator alone raises macro element Top-1 to 0.667; mechanism-specific element interactions recover 1.000. Max-aggregated probabilities are ranking diagnostics rather than calibrated existential probabilities.

Table 9: Full mechanism-aware action-BCE diagnostics. B/E/M denote held-out Broccoli, Expense, and Markor element Top-1.

Training unit	B	E	M	Elem.	Action	Rank	Mass	BCE	Brier
Shared action BCE	0.000	1.000	0.800	0.600	0.300	1.400	0.542	0.158	0.046
Additive action BCE	0.000	1.000	1.000	0.667	0.000	1.333	0.581	0.152	0.047
Interaction action BCE	1.000	1.000	1.000	1.000	0.333	1.000	0.805	0.109	0.030
Element BCE	1.000	1.000	1.000	1.000	–	1.000	0.756	0.193	0.052

B.3 Relational transfer controls

This separate study scores operation and referent matches. The separated head uses operation cosine similarity, referent cosine and surface similarity, two action-type indicators, and candidate center coordinates and area. The interaction variant adds the products of operation similarity with each referent similarity; both heads use standardized, regularized logistic regression. Direct+ tests the comparison under matched low-data conditions.

Table 10: Leave-one-application-out element realization under natural candidates.

Method	Broccoli	Expense	Markor	Macro	Valid mass
Matched direct	0.000	1.000	0.000	0.333	0.179
Direct+	0.250	0.800	0.000	0.350	0.318
Atomic max	0.250	0.333	0.217	0.267	0.110
Separated factors	1.000	1.000	1.000	1.000	0.846
+ continuous interaction	1.000	1.000	1.000	1.000	0.881

B.4 ADDITIONAL STATUS-HEAD RESULTS

The 48-case status-head study has 16/0/32 wins/losses/ties for MAI and 13/0/35 for UI-Venus. In its 24-state development comparison, learning all relations reaches 18/24 because it damages title matching; the targeted status head reaches 24/24 (Table 11).

Additional selector comparisons. Table 11 compares the relation selectors with an earlier whole-instruction FPIR adapter. This baseline applies its fixed nine-feature linear head to the full instruction and card descriptor, using semantic similarity, literal-span identity, the six role flags, and the identity–non-editability interaction. Unlike the field-specific selector, it does not score title, status, and team separately. The status-head variant has 21 wins, 2 losses, and 25 ties against this baseline on the 48 held-out selection cases (exact sign-test $p = 6.60 \times 10^{-5}$). Each sentence structure contributes 20/24 correct predictions.

Table 11: Full relation-level element-selection comparison.

Method	Development	Held-out selection	MAI native	Venus native
Whole-instruction FPIR	11/24	21/48	—	—
Raw MiniLM signed factors	16/24	24/48	26/48	24/48
Learned relations for all fields	18/24	—	—	—
Learned status head	24/24	40/48	42/48	37/48

B.5 NATIVE COORDINATE DIAGNOSTICS AND REPLICATION

Table 12: Earlier strict checker isolation of coordinate-factor repair.

Scope	n	MAI	FPIR-cond.	W/L/T	Δ	Exact p
Broccoli	15	15/15	15/15	0/0/15	0.000	1.000
Markor	15	12/15	15/15	3/0/12	+0.200	0.250
Pooled	30	27/30	30/30	3/0/27	+0.100	0.250

All 60 arms in the 30-state coordinate intervention have a valid candidate, parseable action, and completed execution (Table 13). After the evaluated click or long press, the runner applies the same fixed task-specific continuation in both arms. "Continuation reached" means these steps arrived at the state where the task checker can be applied. The official checker and a stricter target-only predicate agree on every arm.

Table 13: Native execution stages over all 30 matched states per method.

Stage	Frozen MAI	FPIR-conditioned MAI
Valid candidate available	30/30	30/30
Native action parseable	30/30	30/30
Execution completed	30/30	30/30
Fixed continuation reached	27/30	30/30
Strict target-only checker success	27/30	30/30

Nearest-center projection attributes only 16/30 original MAI actions to a valid element, although native execution succeeds on 27/30; conditioned projection is 23/30 while execution is 30/30. Projection is therefore useful for offline attribution but not as a task-success surrogate.

A separately declared 40-seed Markor replication retains one candidate-coverage failure and executes all 39 covered pairs. Unconditioned and conditioned MAI obtain 37/39 and 39/39 strict success (2/0/37; exact sign-test $p = 0.5$). Both gains preserve long press and the target row while moving $x = 79$ to 426 or 428. This replicates the coordinate-repair effect descriptively; its discordant pairs are not pooled with the 30-state estimate.

Reproducibility records. Code and experimental records are available in our anonymous repository. The experiment records retain configurations, prompts, evaluation inputs, parsed actions, native outcomes, and execution commands. Per-state construction seeds and AgentCPM sampling seeds are retained with their corresponding records.