PFP-WORLD: PRE-POLICY PLANNING FOR WEB AGENTS WITH A PAGE FUNCTIONAL PRIMITIVE-BASED WORLD MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model-based web agents struggle with long-horizon cross-page web navigation, as they cannot proactively anticipate environmental dynamics. World models could resolve this by simulating page state transitions in advance, yet existing web world models mainly serve post-hoc purposes: they either generate full trajectories for policy training or evaluate atomic actions derived from flawed subgoal reasoning, lacking a mechanism to screen candidate page-level goals prior to execution. To fill this gap, we propose PFP-WORLD, a pre-policy world-model planner built on Page Functional Primitives (PFPs). PFPs denote page-level functional intentions, serving as unified semantic abstractions that group atomic actions by shared functional intent (e.g., search, checkout). We encode each page state as an inventory of available PFPs. PFP-WORLD redefines three core components of conventional web world models: it shifts the world model's role from a post-hoc trajectory generator or policy evaluator to a pre-policy planner, replaces raw HTML DOM state encoding with abstract PFP inventories, and uses semantic PFPs as high-level control signals for state transition instead of atomic action spaces. With our PFP-World, the planned primitives guide downstream action grounding by mapping each high-level functional intent to executable web operations. We further introduce a page-centric bottom-up paradigm to learn site dynamics from PFP transitions without relying on complete task trajectories. Experiments on the standard WebArena benchmark and a self-constructed cross-page benchmark (namely CrossPageBench) show that PFP-WORLD-SFT outperforms existing world-model-based baselines, bringing a 20.7 percentage-point gain in overall success rate over the reproduced WMA baseline on WebArena and gains of 19.8 and 12.8 percentage points in average target completion over the reproduced RLVR-World baseline on the Shopping and GitLab sites of CrossPageBench, respectively.

1 INTRODUCTION

Web agents aim to automate multi-step tasks on web platforms, from e-commerce transactions and form filling to cross-site information retrieval (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024; Koh et al., 2024; Zheng et al., 2024a; He et al., 2024; Yang et al., 2025; Drouin et al., 2024; Yoran et al., 2024). Yet agents remain unreliable on long-horizon goals that span multiple pages (Li & Waldo, 2024; Shlomov et al., 2025; Ye et al., 2026; Wu et al., 2025b; Jang et al., 2026). Before generating the next atomic action, an agent must first make a page-level decision about what should be accomplished next on the current page, rather than how to carry it out through concrete actions. World models offer a natural mechanism for comparing possible page-level decisions by predicting how each would change the current page before execution (Chae et al., 2025; Gu et al., 2025; Wu et al., 2025a; Mei et al., 2026; Xiao et al., 2026).

Existing world-model-augmented pipelines assist web-agent policies in two main ways: by using the world model as a trajectory generator (Fang et al., 2025; Gao et al., 2026) or as a post-hoc policy evaluator (Wu et al., 2025a; Chae et al., 2025). In the first role, the world model produces simulated trajectories as experience for policy training. Because errors can compound over multi-step rollouts, unreliable simulated trajectories may provide misleading supervision for the policy. In the second

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.

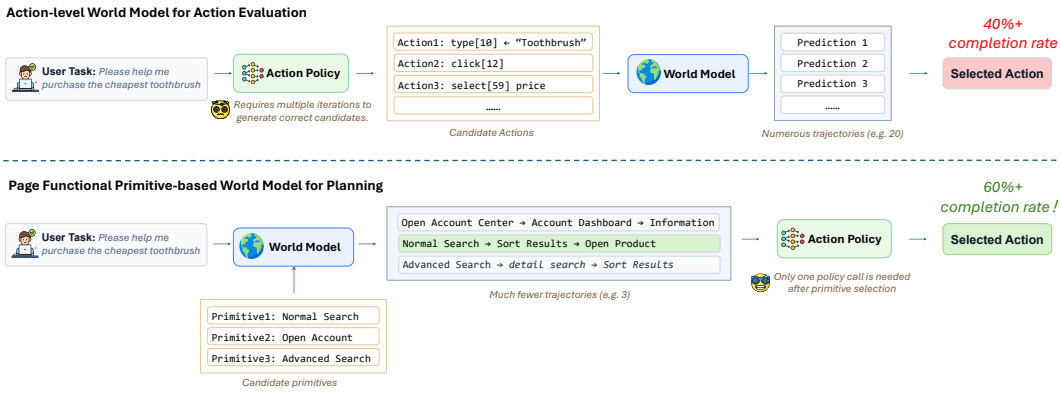


Figure 1: Existing action-level web world models commonly evaluate atomic-action candidates after they have been generated by the policy. PFP-WORLD instead uses the world model as a pre-policy planner over page functional primitives and provides the selected primitive as the current plan before the policy generates atomic actions.

role, the world model predicts the outcomes of atomic-action candidates proposed by the policy and uses those predictions to verify or rank the candidates. This design therefore relies on the policy to include a potentially correct action in the candidate set. *Therefore, defining and building world models that effectively assist web agent policy models remains a challenging task.*

We address this challenge by rethinking where the world model intervenes and what it predicts. We introduce PFP-WORLD, a pre-policy world-model planner built on Page Functional Primitives (PFPs). A PFP represents a candidate page-level goal as a functional intent, grouping entity-specific actions and multi-action realizations that accomplish the same page function. Using PFPs, we represent each page state by its inventory of available PFPs rather than a page-specific DOM or screenshot, and use a selected PFP rather than an atomic action as the control signal for state transition. As shown in Figure 1, PFP-World shifts the world model's role from a policy evaluator to a pre-policy planner. This formulation allows the world model to predict how a page-level goal changes the available page functions before the action policy grounds it into concrete operations, reducing the policy model's reasoning burden. Moreover, we observe that the PFP representation is more stable than DOM element-based representation by conducting a cross-version comparison of 18 matched page snapshots from two production websites: only 29.8% of interactive page elements overlap between the 2024 and 2026 versions, compared with 81.54% of PFPs (Appendix B).

At planning time, the user goal guides PFP selection, while the site dynamics predict the successor primitive inventories. Alternating selection and prediction produces multi-step PFP trajectories whose consequences are compared before atomic-action generation. PFP-WORLD provides the parameterized first PFP of the selected trajectory to the action policy as the current plan, after which the policy grounds it into executable browser actions. This division of labor moves page-level planning into the world model and narrows the downstream policy's role from deciding both *what* to do and *how* to do it to grounding *how* to execute the selected plan.

To facilitate the construction and training of the world model, we propose a page-centered, bottom-up data collection paradigm. Compared with task-centric full-trajectory collection (Fan et al., 2026; Xiao et al., 2026), our data collection does not rely on full task trajectories. Instead, it enumerates individual primitive transitions and records how each primitive changes the page state. PFPs reduce the number of distinct operations to model by grouping entity-specific actions and multi-action realizations, while page types allow local transitions to be reused across functionally similar pages. This local collection reduces data collection complexity and expands coverage over page types and primitive transitions. Moreover, because the resulting local dynamics are not tied to a particular goal, they can be reused across tasks. These data directly support retrieval and supervised fine-tuning, while a prompted variant identifies inventories and predicts transitions without parameter updates.

We evaluate the framework on four sites from the standard WebArena benchmark (Zhou et al., 2024) and on CrossPageBench, our benchmark for cross-page, multi-stage tasks on Shopping and GitLab. On WebArena, PFP-WORLD-SFT achieves a 36.9% overall success rate, compared with 16.2% for

the reproduced WMA baseline. On CrossPageBench, the same SFT variant achieves average target completion of 64.5% and 60.3% on Shopping and GitLab, respectively, compared with 44.7% and 47.5% for the reproduced RLVR-World baseline. Without transition retrieval or parameter updates, PFP-WORLD-PROMPT reaches 62.7% and 63.0% on the two sites and exceeds Plan–Action–Reflect and Agent Workflow Memory under the same backbone.

In summary, our main contributions are:

- We reposition the world model as a pre-policy planner. It builds on page functional primitives to screen candidate page-level goals and reduce the policy model's reasoning burden.
- We design functional primitive-based page states and control signals, instead of raw visual/DOM observations and atomic actions, enhancing stability across visually distinct yet functionally similar web pages.
- We propose a bottom-up, page-level data collection paradigm for world modeling without relying on complete task trajectories, which reduces data collection complexity and facilitates cross-task usage.

2 RELATED WORK

Web and GUI agents. Web agents have progressed from controlled benchmarks (Yao et al., 2022; Deng et al., 2023) to realistic browser environments with persistent state and visually grounded tasks (Zhou et al., 2024; Koh et al., 2024; Zheng et al., 2024b). Recent methods improve visual and HTML-aware grounding, multimodal inputs, and observation/action spaces (Zheng et al., 2024a; Cheng et al., 2024; Hong et al., 2024; He et al., 2024; Furuta et al., 2024; He et al., 2025; Yang et al., 2025). These agents typically execute atomic actions or short action sequences, guided by reasoning and reflection mechanisms such as ReAct and Reflexion (Yao et al., 2023; Shinn et al., 2023) and sometimes augmented with search over real environment outcomes (Koh et al., 2025). Hierarchical agents instead plan above atomic actions (Gur et al., 2024): WebPilot executes branches in the live browser (Zhang et al., 2025), SteP uses hand-authored site handlers (Sodhi et al., 2024), and Agent Workflow Memory (AWM) retrieves past workflows (Wang et al., 2025). Our planner also plans above atomic actions, but simulates transitions over functional primitives without executing candidate branches in the live environment.

World models for web agents. Predicted futures can improve web-agent decisions (Chae et al., 2025; Gu et al., 2025; Wu et al., 2025a; Mei et al., 2026; Xiao et al., 2026; Shen et al., 2026). WMA predicts natural-language next-state deltas to score atomic action candidates (Chae et al., 2025); WebDreamer combines LLM world models and value functions to score their outcomes (Gu et al., 2025); and RLVR-World trains tokenized state prediction with verifiable rewards (Wu et al., 2025a). WebWorld scales multi-format, long-horizon simulation to more than one million open-web trajectories for agent training and inference-time search (Xiao et al., 2026). These systems keep simulated transitions close to the observation/action stream, pairing DOM or page-state tokens with low-level actions, and use them to train a policy or evaluate its candidates (Chae et al., 2025; Gu et al., 2025; Shen et al., 2026). We instead simulate PFPs before policy invocation, changing both the granularity and point of intervention. This makes explicit a prior: large sites reuse page templates and workflows across URLs, allowing reusable page types to index their available functional primitives (Sec. 3.1; Appendix B) (Deng et al., 2023; Zhou et al., 2024; Yang et al., 2025).

Synthetic and recreated web environments. A complementary line constructs executable environments for scalable agent training. AutoWebWorld generates verifiable interactive websites from explicit finite-state machines (Wu et al., 2026). VeriEnv recreates existing websites and exposes controlled internal state through an SDK, enabling generated tasks with deterministic rewards (Chae et al., 2026). These frameworks create environments for policy learning; our method instead models site-local dynamics over functional primitives and queries them during planning.

3 WORLD-MODEL DATA CONSTRUCTION

We construct data for PFP-World using a page-centered, bottom-up procedure whose basic unit is a PFP transition. Because collection is independent of benchmark goals, the data can be reused across tasks.

3.1 PAGE FUNCTIONAL PRIMITIVE INVENTORIES

Functional page types. Large websites often contain functionally equivalent pages at different URLs despite different entities and rendered layouts. Modeling each URL separately would duplicate equivalent states and transitions. We group such pages into a functional page type T.

Primitive inventories. For each type T, we define a primitive inventory $\mathcal{I}(T)$ shared by pages of that type. Each $\tau \in \mathcal{I}(T)$ denotes an available PFP. During rollout, PFP slots remain unbound; TRAJECTORY SELECTION fills the slots of the selected first PFP for execution. A PFP is atomic in the semantic control space of the world model, although the action policy may execute it through one or more atomic actions. It is also page-closed: execution remains on the current page, except that the final action may navigate to another page. The planner treats the inventory as the current state and a selected PFP as the control signal to that state.

Human reviewers merge or split candidate page groups and validate their inventories proposed by a multimodal model, since structural or visual similarity does not guarantee functional equivalence. Each validated group defines a functional page type and its inventory (Appendix B).

We use construction-time statistics to examine whether PFPs abstract beyond atomic actions while remaining executable. On Shopping, a PFP requires 1.80 atomic actions on average, and 38.95% require at least two. It also groups object-specific actions with the same functional meaning: one category page contains 90 concrete actions but only 17 PFPs; for example, twelve product links instantiate the same "view any product details" primitive. The world model can therefore predict the product-detail inventory without first specifying a particular product. During collection, the browser worker executes each proposed PFP as a local instruction, and 92.1% of the resulting action trajectories pass human validation.

3.2 PAGE-CENTERED CONSTRUCTION OF SITE DYNAMICS

A primitive inventory describes what can be done in the current page state, but planning also requires knowing how each PFP changes that state. During collection, for each available PFP, we fill its slots with valid page-specific entities or values, execute it, and record the source type T, the PFP τ, the resulting type T', and any within-type state update. We label the transition a *type change* when $T' \neq T$ and *within type* otherwise, independently of browser navigation. Together, these records provide reusable local dynamics used during planning. Unlike task-centric full-trajectory collection (Fan et al., 2026; Fang et al., 2025; Gao et al., 2026; Xiao et al., 2026), our coverage unit is a reusable page-type–primitive transition rather than a benchmark goal or complete solution path.

Collection procedure. Crawling from the homepage expands through new page types and reuses inventories of covered types; human reviewers validate all admitted page groups, inventories, and transitions (Appendix B).

Data scope. We collect 339 functional page types and 5,766 transition records from the WebArena mirror sites Shopping, CMS, GitLab, and Reddit. Collection is independent of benchmark goals, and records contain no WebArena or CrossPageBench goal, checkpoint, or task identifier. Appendix B gives the collection protocol, record schema, interaction scope, and per-site statistics.

4 METHOD

Given the current observation o_t, user goal g, and execution history, WORLD MODEL PLANNING returns the selected trajectory's parameterized first PFP for ACTION GROUNDING to execute (Figure 2). DUAL JUDGE directs execution retries or task-level judgment; unfinished tasks then return to planning with the resulting observation and PFP-level feedback.

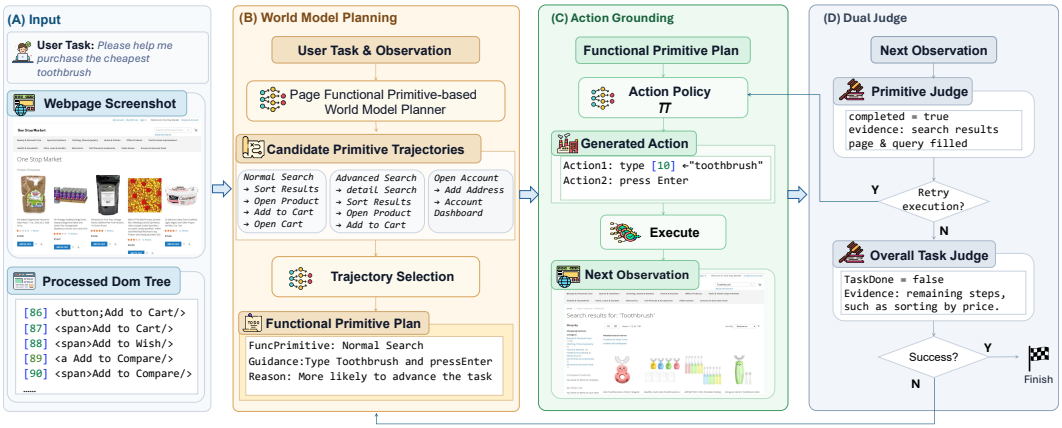


Figure 2: Pipeline overview. INPUT provides the current page observation; PFP-WORLD maps it to a primitive inventory, rolls out primitive trajectories to a fixed depth, and provides the selected trajectory's parameterized first PFP as the current plan. ACTION GROUNDING executes the plan. DUAL JUDGE decides whether to retry execution or proceed to task-level judgment. Unfinished tasks return to planning with new observations and PFP-level feedback.

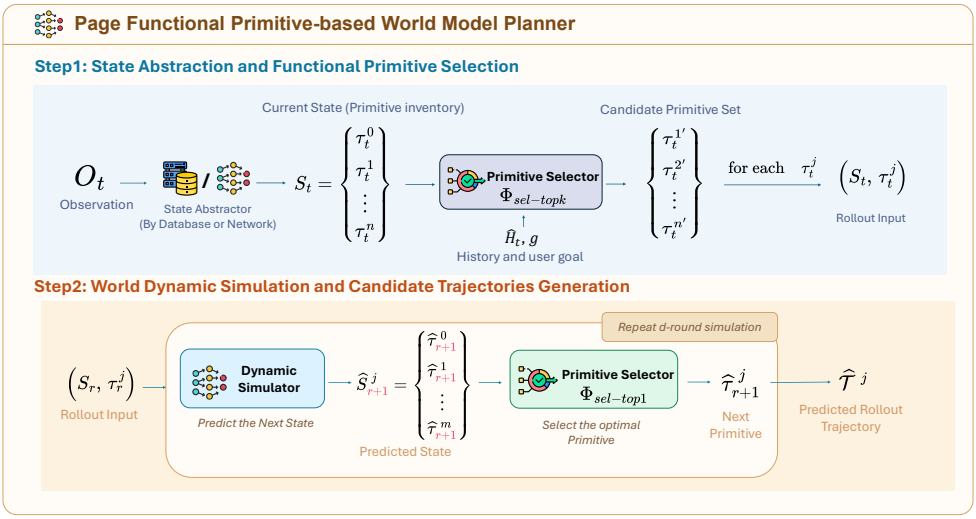


Figure 3: World-model planning in PFP-WORLD. Each branch j starts from $\widehat{S}_0^j = S_t$, the inventory produced by STATE ABSTRACTOR. PRIMITIVE SELECTOR uses the goal, inventory, and branch history; DYNAMIC SIMULATOR predicts the outcome from the inventory and selected PFP. The planning loop appends the PFP and predicted effects to branch history. The lower panel shows a rollout step from r to $r+1$, repeated to form three fixed-depth trajectories.

4.1 WORLD MODEL PLANNING

WORLD MODEL PLANNING constructs a PFP-level plan before atomic-action generation. STATE ABSTRACTOR maps the page to its primitive inventory. PRIMITIVE SELECTOR and DYNAMIC SIMULATOR alternate to build goal-conditioned, fixed-depth trajectories. Before atomic-action generation, TRAJECTORY SELECTION evaluates their predicted multi-step consequences and parameterizes the selected trajectory's first PFP as the current plan.

4.1.1 STATE ABSTRACTOR

STATE ABSTRACTOR, the only planning module that reads the raw page, maps o_t to the abstract state $S_t = \mathcal{I}(T_t)$, where T_t is the functional page type. Pages of the same type share this inventory

despite differences in URLs, entities, or interface details; execution and branch histories supply the context for choosing among its PFPs.

4.1.2 PRIMITIVE SELECTOR

PRIMITIVE SELECTOR chooses the next PFP using the user goal g, current abstract state $\widehat{S}_r$, and branch history $\widehat{H}_r$. History records prior primitives and their outcomes, including within-type updates, so choices can differ even when the inventory remains unchanged.

Let H_t record the details and observed outcomes of prior PFP executions. Each candidate branch starts from $\widehat{S}_0 = S_t$ and initializes its history as $\widehat{H}_0 = H_t$. At rollout step r, the selector returns

$$\tau_r \in \text{Select}\left(g, \widehat{S}_r, \widehat{H}_r\right). \tag{1}$$

The selector returns three root candidates and one continuation per branch thereafter. Each selected τ_r remains a general PFP with unbound slots and serves as the control signal for transition prediction.

History for subsequent selection. After DYNAMIC SIMULATOR predicts a transition, the planning loop updates that branch's history independently:

$$\widehat{H}_{r+1} = \widehat{H}_r \parallel \widehat{h}_r. \tag{2}$$

Here, $\widehat{h}_r$ records the selected general PFP and its effects, including any within-type update, and $\parallel$ denotes appending a record. The successor inventory is carried forward separately as the next rollout state, while PRIMITIVE SELECTOR uses the updated history to choose the next PFP.

4.1.3 DYNAMIC SIMULATOR

DYNAMIC SIMULATOR predicts a transition conditioned only on the current primitive inventory and selected PFP:

$$\left(\widehat{S}_{r+1}, \ell_r, \Delta_r\right) = \text{Sim}\left(\widehat{S}_r, \tau_r\right). \tag{3}$$

Here, $\widehat{S}_{r+1}$ is the predicted successor state, ℓ_r the transition type, and Δ_r any within-type update. Histories guide primitive selection, indirectly determining the simulated operation.

A type change yields another functional page type's inventory; a within-type transition retains the inventory and returns the page-internal update. The transition type depends on whether the functional page type changes, independently of browser navigation (Sec. 3.2). Repeating selection and prediction to depth d produces one trajectory per root candidate. In the Retrieval variant, the page type T is retained as internal lookup metadata.

4.1.4 TRAJECTORY SELECTION

Given the user goal, execution history, and candidate trajectories, TRAJECTORY SELECTION chooses the rollout that best advances the task and fills the slots of its first PFP with the required entities or values. The resulting parameterized PFP, denoted $\gamma^\star$, is sent to ACTION GROUNDING. Only this first PFP is executed; the remaining rollout is used to evaluate it.

4.2 ACTION GROUNDING AND DUAL JUDGE

ACTION GROUNDING converts the parameterized PFP $\gamma^\star$ into atomic browser actions using the current observation o_t and user goal g. After execution, DUAL JUDGE checks the outcome. When the selected PFP remains appropriate but execution fails, ACTION GROUNDING retries the current plan. Other outcomes proceed to task-level judgment. A completed task ends the run; otherwise, the outcome updates the real history, and the new observation starts the next planning cycle.

4.3 IMPLEMENTATION VARIANTS

The three variants share the complete planning and execution loop. *Retrieval* uses the collected inventory index and retrieves recorded transitions without parameter updates.

SFT fine-tunes STATE ABSTRACTOR and DYNAMIC SIMULATOR on the same data; PRIMITIVE SELECTOR remains unmodified. *Prompt* implements STATE ABSTRACTOR, PRIMITIVE SELECTOR, and DYNAMIC SIMULATOR through prompting alone, without collected world-model data or parameter updates. Appendix D details retrieval and training.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Evaluation suites and metrics. Our main comparisons use two complementary suites. *WebArena* measures end-to-end success on Shopping, CMS, GitLab, and Reddit under its official task definitions and metric (Zhou et al., 2024). To evaluate cross-page, multi-stage progress directly, we construct *CrossPageBench*, comprising 62 source templates and 347 goal instances on Shopping and GitLab. Its tasks are designed around user scenarios, independently of the collected PFP transition data. Each task spans at least two functional page types. Because these goals admit meaningful partial progress, we report target completion: the per-task fraction of achieved checkpoints averaged across tasks. Appendix A provides the construction protocol, statistics, and examples.

Compared methods. We compare our pre-policy formulation with WMA (Chae et al., 2025) and RLVR-World (Wu et al., 2025a), which predict transitions from DOM-based observations and atomic actions. We run their released implementations, replacing the original policy backbone with `gpt-5.2`. We also compare with PLAN–ACTION–REFLECT, a planning/action/reflection harness, and Agent Workflow Memory (AWM) (Wang et al., 2025), which retrieves prior workflows; both use the same backbone.

PFP-WORLD variants and shared settings. We evaluate the three implementation variants in Sec. 4.3, with model assignments detailed in Appendix F. *Prompt* uses `gpt-5.2` for all three planning modules without collected world-model data or parameter updates. Across variants, TRAJECTORY SELECTION, ACTION GROUNDING, and DUAL JUDGE use `gpt-5.2`. The default rollout depth is d=3, with three root candidates and one continuation per branch thereafter.

Data separation and execution protocol. For WebArena and CrossPageBench, training or retrieval data for WMA, RLVR-World, Retrieval, and SFT come from the same four WebArena mirror sites in each method's native representation. They contain no CrossPageBench goals or task identifiers. We run every benchmark task three times under the same browser-runner setting and report its highest score across the three executions.

5.2 MAIN RESULTS

WebArena. Both Retrieval and SFT more than double the overall success rate of the reproduced WMA and RLVR-World baselines (Table 1). Each variant also exceeds both baselines on all four sites, with gains of 11.7–28.7 percentage points over the stronger baseline on each site.

Table 1: End-to-end success rate (%) on four WebArena sites. † denotes WMA (Chae et al., 2025) and RLVR-World (Wu et al., 2025a) reproduced with `gpt-5.2` as the policy model.

Method	Shopping	CMS	GitLab	Reddit	Overall
Baselines					
WMA†	19.3%	14.8%	16.1%	13.2%	16.2%
RLVR-World†	21.4%	10.4%	11.7%	11.3%	14.1%
Ours					
PFP-WORLD-RETRIEVAL	33.1%	38.0%	36.1%	26.4%	34.0%
PFP-WORLD-SFT	35.3%	42.4%	29.4%	41.9%	36.9%

CrossPageBench. Table 2 shows that both variants outperform both action-level world-model baselines on Shopping and GitLab. Relative to RLVR-World, the stronger baseline in both columns,

Table 2: Average CrossPageBench target completion (%), computed per task as the fraction of achieved checkpoints and then averaged across tasks.

Method	Shopping	GitLab
Baselines		
WMA (Chae et al., 2025)	42.4%	45.1%
RLVR-World (Wu et al., 2025a)	44.7%	47.5%
Ours		
PFP-WORLD-RETRIEVAL	62.0%	61.2%
PFP-WORLD-SFT	64.5%	60.3%

Table 3: Average target completion (%) on CrossPageBench for planning methods using `gpt-5.2` as the backbone.

Method	Shopping	GitLab
PLAN–ACTION–REFLECT	49.9%	47.3%
AWM (Wang et al., 2025)	46.8%	50.3%
PFP-WORLD-PROMPT	62.7%	63.0%

Table 4: Average target completion (%) when varying the dynamics implementation and rollout depth on the same CrossPageBench Shopping subset. Each d=0 row uses its variant's state abstractor and remaining model configuration, with transition simulation disabled.

Planning configuration	Variant	Depth d	Shopping
Direct PFP selection	Retrieval	0	52.8%
PFP selection + rollout	Retrieval	1	54.2%
PFP selection + rollout	Retrieval	3	63.8%
PFP selection + rollout	Retrieval	5	64.1%
Direct PFP selection	SFT	0	52.8%
PFP selection + rollout	SFT	1	53.9%
PFP selection + rollout	SFT	3	68.9%
PFP selection + rollout	SFT	5	65.7%

the gains range from 12.8 to 19.8 percentage points. Retrieval and SFT exchange rank across sites, showing gains with both dynamics implementations.

Comparison with planning methods. To evaluate the functional-primitive planning interface without retrieved or fine-tuned dynamics, we compare PFP-WORLD-PROMPT with two planning methods in Table 3. All use `gpt-5.2`; PLAN–ACTION–REFLECT also shares our ACTION GROUNDING but replaces upstream planning, whereas AWM retains its workflow-memory design.

PFP-WORLD-PROMPT exceeds the stronger comparison method by 12.8 and 12.7 percentage points on Shopping and GitLab. Its gains over PLAN–ACTION–REFLECT, which shares the same backbone and ACTION GROUNDING, support the value of the functional-primitive planning interface.

5.3 ABLATIONS

We first examine the contribution of predictive rollout on a randomly sampled CrossPageBench Shopping subset. At d=0, each variant retains its state abstractor and other model settings, selects a PFP directly, and performs no transition simulation. Both Retrieval and SFT achieve 52.8% under this direct-selection setting. At positive depths, each uses its corresponding dynamics implementation.

Depth 1 offers modest gains over direct PFP selection, whereas depth 3 improves target completion by 11.0 and 16.1 percentage points with Retrieval and SFT, respectively. The 9.6- and 15.0-point improvements from depth 1 to 3 show the benefit of considering multi-step consequences. Depth 5 provides no consistent gain.

We evaluate DUAL JUDGE and its next-loop feedback on the full CrossPageBench Shopping task set using the SFT variant at depth 3. Removing this feedback path lowers target completion from 64.5% to 59.8%, a decrease of 4.7 percentage points (Appendix D). The checks and feedback are ablated together because the checks produce the feedback used by the next planning cycle.

Predictive planning across implementations. The depth ablations show that predictive rollout adds value beyond direct PFP selection while retaining the PFP representation and each variant's model configuration (Table 4). The Prompt comparison provides complementary evidence: PFP-WORLD-PROMPT outperforms PLAN–ACTION–REFLECT with the same backbone and ACTION GROUNDING, without collected world-model data or parameter updates (Table 3). Together with the Retrieval and SFT task results, these comparisons support predictive planning over PFPs across retrieved, fine-tuned, and prompted dynamics (Tables 1 and 2; Appendix D).

Model usage and runtime. At depths 1 and 3, PFP-WORLD-SFT uses fewer model calls and tokens than WMA but has longer client-observed runtime per matched control-loop iteration (Appendix F.1). Task-level compute also depends on the number of iterations.

5.4 COMPONENT AND TRANSFER ANALYSIS

We first assess the two predictive planning modules. STATE ABSTRACTOR reaches 92.49% exact match when mapping a page to its complete primitive inventory, with inventory order ignored. DYNAMIC SIMULATOR reaches 91.96% semantic-match accuracy when predicting the resulting primitive inventory after one PFP. Appendix B further examines the within-type consistency and cross-version stability of the PFP representation.

Few-shot cross-site adaptation. We compare Qwen3-VL-8B-Instruct and WebArena-trained DYNAMIC SIMULATOR initializations for target-site transition prediction using identical target-site support transitions and optimization settings. Evaluation covers a template-disjoint Amazon split and a pool of Practice Toolshop, DemoBlaze, and ExpandTesting Bookstore, none of which appears in the source dynamics training data (Appendix C). Transition-State F1 measures transition type, primitive additions and removals, and within-type updates, excluding unchanged primitives.

Figure 4 in Appendix C reports the complete support curves. WebArena initialization improves Transition-State F1 at all six positive-support budgets on Amazon and all three in the pooled panel. With $K{=}8$ labeled support transitions per site, F1 rises from 32.42% to 72.93% on Amazon and from 53.35% to 61.54% on the three-site pool. On Amazon, the WebArena-initialized model with eight support transitions also exceeds the base model trained on all 56 support transitions (49.56% F1). These results show the value of learned dynamics for prediction with limited target-site support. The per-site results in Appendix C show variation around this aggregate advantage: DemoBlaze favors Base at full support, and the seven-transition Recreation.gov test yields mixed results.

End-to-end cross-site evaluation. We further evaluate PFP-WORLD-SFT after target-site adaptation with Full support on 10 user tasks each from Amazon and Practice Toolshop. It achieves 75.3% average target completion, compared with 71.4% for PLAN–ACTION–REFLECT, a gain of 3.9 percentage points (Appendix F.2).

6 CONCLUSION

We introduced PFP-WORLD, a pre-policy world-model planner that represents web states as inventories of Page Functional Primitives (PFPs) and uses PFPs as high-level control signals. By simulating the multi-step consequences of candidate PFPs before atomic-action generation, it determines which page-level intention to execute before grounding how to execute it. We further construct site dynamics through a page-centered, bottom-up procedure over reusable PFP transitions rather than complete task trajectories.

Experiments show consistent gains over reproduced world-model baselines on both WebArena and CrossPageBench, while the prompted variant also outperforms planning methods on CrossPageBench. Depth ablations show that multi-step rollout improves over direct PFP selection. Few-shot adaptation further shows that the learned dynamics provide a stronger initialization for target-site transition prediction. End-to-end experiments demonstrate PFP-WORLD's cross-site generalization after adapting its dynamics to target websites. Together, these results support functional primitives as a predictive state and control space, and pre-policy rollout as an effective source of guidance for cross-page web agents.

AI USE STATEMENT

Generative AI tools were used for language polishing, feedback on methodological descriptions and interpretation of results, and assistance with PFP data construction, evaluation-checkpoint construction, and data review. The model-assisted data and checkpoint procedures are described in Sec. 3 and Appendices A and B. The authors reviewed the AI-assisted revisions, and human reviewers validated the collected page types, inventories, and transitions. The authors take responsibility for the final content, claims, and research artifacts.

ETHICS STATEMENT

The experiments use WebArena mirror environments, public web pages, and test or researcher-controlled accounts and data. No other users' private data were used, and no real payments or orders were placed. Because web agents can change account and transaction state, deployment should require user authorization and appropriate safeguards for consequential actions.

REPRODUCIBILITY STATEMENT

Sections 3–4 describe the data construction and planning method, and Sec. 5 specifies the experimental comparisons and metrics. Appendices A–B describe the benchmark and PFP data; Appendix C reports the cross-site prediction protocol and full support curves. Module interfaces and training mappings are provided in Appendix D, with model assignments, execution and reporting settings, and end-to-end cross-site evaluation in Appendix F. Anonymous code and data are provided as supplementary materials.

REFERENCES

Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a00548031e4647b13042c97c922fadf1-Abstract-Conference.html.

Hyungjoo Chae, Jungsoo Park, and Alan Ritter. Safe and scalable web agent learning via recreated websites. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/60744.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.505/.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Sicheng Fan, Rui Wan, Yifei Leng, Gaoning Liang, Li Ling, Yanyi Shang, and Dehan Kong. WebChain: A large-scale human-annotated dataset of real-world web interaction traces, 2026. URL https://arxiv.org/abs/2603.05295.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. WebEvolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8959–8975, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.454. URL https://aclanthology.org/2025.emnlp-main.454/.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/7ef7d8359036afd8c2378d82c21058a4-Abstract-Conference.html.

Yifei Gao, Junhong Ye, Yifan Yang, Jiaqi Wang, Yi Zhang, Zhang Ruichen, and Jitao Sang. WebSynthesis: World model-guided monte carlo tree search for efficient WebAgent trajectory synthesis. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25241–25258, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1157. URL https://aclanthology.org/2026.acl-long.1157/.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=c6l7yA0HSq.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world WebAgent with planning, long context understanding, and program synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9JQtrumvg8.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.371/.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. OpenWebVoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://aclanthology.org/2025.acl-long.1336/.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Hong_CogAgent_A_Visual_Language_Model_for_GUI_Agents_CVPR_2024_paper.html.

Lawrence Keunho Jang, Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov. Odysseys: Benchmarking web agents on realistic long horizon tasks, 2026. URL https://arxiv.org/abs/2604.24964.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.50/.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=QF0N3x2XVm.

Eric Li and Jim Waldo. WebSuite: Systematically evaluating why web agents fail, 2024. URL https://arxiv.org/abs/2406.01623.

Kai Mei, Jiang Guo, Shuaichen Chang, Mingwen Dong, Dongkyu Lee, Xing Niu, and Jiarong Jiang. R-WoM: Retrieval-augmented world model for computer-use agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5ZaoXB3MdP.

Zhouzhou Shen, Xueyu Hu, Xiyun Li, Tianqing Fang, Juncheng Li, and Shengyu Zhang. World-model-augmented web agents with action correction, 2026. URL https://arxiv.org/abs/2602.15384.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Segev Shlomov, Ben Wiesel, Aviad Sela, Ido Levy, Liane Galanti, and Roy Abitbol. From grounding to planning: Benchmarking bottlenecks in web agents. In *ECAI 2025 - 28th European Conference on Artificial Intelligence, including 14th Conference on Prestigious Applications of Intelligent Systems, PAIS 2025 - Proceedings*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 4815–4822, 2025. doi: 10.3233/FAIA251390. URL https://doi.org/10.3233/FAIA251390.

Paloma Sodhi, S. R. K. Branavan, Yoav Artzi, and Ryan McDonald. SteP: Stacked LLM policies for web actions. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=5fg0VtRxgi.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Jialong Wu, Shaofeng Yin, Ningya Feng, and Mingsheng Long. RLVR-World: Training world models with reinforcement learning. In *Advances in Neural Information Processing Systems 38*, pp. 138910–138948, 2025a. doi: 10.52202/085713-4177. URL https://doi.org/10.52202/085713-4177.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. WebWalker: Benchmarking LLMs in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10290–10305, 2025b. doi: 10.18653/v1/2025.acl-long.508. URL https://aclanthology.org/2025.acl-long.508/.

Yifan Wu, Yiran Peng, Yiyu Chen, Jianhao Ruan, Zijie Zhuang, Cheng Yang, Jiayi Zhang, Man Chen, Yenchi Tseng, Zhaoyang Yu, Liang Chen, Yuyao Zhai, Bang Liu, Chenglin Wu, and Yuyu Luo. AutoWebWorld: Synthesizing infinite verifiable web environments via finite state machines. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/62161.

Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin, and Zuozhu Liu. WebWorld: A large-scale world model for web agent training. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/65352.

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

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Suyu Ye, Haojun Shi, Darren Shih, Hyokun Yun, Tanya G. Roosta, and Tianmin Shu. RealWebAssist: A benchmark for long-horizon web assistance with real-world users. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34441–34449, 2026. doi: 10.1609/aaai.v40i40.40742. URL https://ojs.aaai.org/index.php/AAAI/article/view/40742.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. AssistantBench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://aclanthology.org/2024.emnlp-main.505/.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. WebPilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23378–23386, 2025. doi: 10.1609/aaai.v39i22.34505. URL https://doi.org/10.1609/aaai.v39i22.34505.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, 2024a. URL https://proceedings.mlr.press/v235/zheng24e.html.

Boyuan Zheng, Boyu Gou, Scott Salisbury, Zheng Du, Huan Sun, and Yu Su. WebOlympus: An open platform for web agents on live websites. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2024b. URL https://aclanthology.org/2024.emnlp-demo.20/.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A CrossPageBench Benchmark Details

CrossPageBench evaluates web agents on user goals that connect several activities across pages. Shopping tasks combine operations such as finding products, comparing alternatives, and arranging a purchase. GitLab tasks combine project setup, content editing, and collaboration. Completing these goals requires coordinating the stages and carrying their effects forward.

Task construction. We construct CrossPageBench tasks around these user scenarios on the Shopping and GitLab sites, independently of the collected PFP transition data. A parameterized source template specifies a task family, and slot bindings produce concrete natural-language goal instances. Each task stays within one site and has machine-checkable completion checkpoints.

Task scale and structure. Table 5 summarizes the 62 source templates and 347 goal instances. Each task spans at least two functional page types and connects multiple stages toward a user goal.

Table 5: CrossPageBench summary statistics by site. Source templates define task families; goal instances bind their slots.

Site	Source templates	Goal instances
Shopping	42	108
GitLab	20	239
Total	62	347

Representative user tasks. Table 6 gives shortened goal instances. The Shopping examples connect product selection with comparison, delivery, and purchase. The GitLab examples connect repository or group creation with subsequent editing and verification. Evaluation uses the complete goals.

Table 6: Representative CrossPageBench task instances, shortened for space. The evaluation uses the full slot-filled natural-language goals.

Site	ID	Shortened goal instance
Shopping	B-01-E1	Create a new U.S. shipping address for Emma Wilson; search for hiking shoes; sort by descending relevance; open the first result; buy it for that address with size 10 and the default color.
Shopping	B-02-E1	Search for Adidas running shoes; sort by descending relevance; add the first two results to comparison; purchase the selected items to a saved address with shoe size 10.
GitLab	G-01-H1	Create an internal project initialized with a README; set the project description; create two branches; add backend and urgent labels; create an issue with both labels; verify it in the issue list.
GitLab	G-02-H2	Create a group and set its description; create a project; create a feature branch; edit `README.md`; commit the change; open and verify a merge request.

Checkpoints and scoring. For each task, we collect a complete reference execution to construct evaluation checkpoints, separately from the local world-model transitions in Sec. 3.2. A model reads the user goal and this trajectory to derive assertions for intermediate achievements and final effects. State assertions check persisted environment state, URL assertions check page-type-scoped URL patterns, and DOM assertions check canonicalized rendered text. These assertions assess task progress without requiring the agent to reproduce the reference trajectory exactly.

Target completion is the fraction of a task's checkpoints that are achieved. We report the mean of the best score for each task, following the repeated-run procedure in Appendix F. It captures partial progress on multi-stage goals, whereas end-to-end success measures whether a task is completed

under its benchmark's success criterion. The CrossPageBench tables report target completion; the WebArena tables report end-to-end success rate.

B Page-Type Data and Representation Stability

Functional page types and PFP inventories. Pages with similar functions share a type T and a common inventory $\mathcal{I}(T)$. The inventory groups operations by functional intent across different entities and page instances. A general PFP supplies the control signal for transition modeling, while its parameterized form specifies the entities or values required for execution.

General primitives and execution examples. Table 7 illustrates this relation for four page types. For example, opening a merge request is a general function whose request slot identifies the object to open. During rollout, the primitive retains its general form; Trajectory Selection fills the slots of the selected first PFP, and Action Grounding executes that instruction. The examples use explicit slot notation to make this relation visible.

Table 7: General PFPs and illustrative execution bindings for representative page types. Slot notation is used to show the relation between a functional primitive and its parameterized execution.

Site / page type	PFP schema	Example parameter
Shopping: Order Details Page	Add {*item*} from the current order to the cart	*item*: "Wireless Mouse"
CMS: Config: Dev	Expand the {*section*} settings	*section*: "Caching"
GitLab: Merge Requests List	Open merge request {*request*}	*request*: "Fix login redirect"
Reddit: Forum Listing	Search site content for {*query*}	*query*: "world models"

Local transition records. Each page-type record contains a label and a representative URL. Local transition records associate a source type and canonical PFP with the resulting type and any page-internal update. Start and terminal URLs record browser navigation. The abstract transition is a type change when $T' \neq T$ and a within-type transition otherwise. Retrieval uses the source type to obtain $\mathcal{I}(T)$ and the selected PFP to obtain its recorded outcome. A within-type transition retains the type-level inventory and returns any page-internal update with the recorded outcome.

Page-centered collection. An automated crawl starts from the homepage and records screenshots, DOMs, and reachable pages. URL patterns and structural or visual similarity suggest groups, and a multimodal model proposes a functional page type and PFP inventory for each group. Human reviewers refine the groups and their inventories. An AI browser worker instantiates and executes each PFP, recording its local action sequence, resulting observation, navigation metadata, and resulting type and update. Terminal URLs extend the crawl frontier after URL normalization; pages of a covered type reuse its inventory, while newly encountered types extend the collection. Reviewers validate the page types, inventories, and transitions admitted to the data. No external crowdworkers were used.

The collection covers Shopping, CMS, GitLab, and Reddit, whose dominant operations admit discrete type-change or within-type transitions. OpenStreetMap's continuous pan and zoom operations change a map viewport incrementally and are outside the current collection scope.

Dataset scale. The collection contains 339 functional page types and 5,766 local transition records. Table 8 reports their distribution across the four sites, separating type changes from within-type transitions.

Within-type functional consistency. We collect DOMs and screenshots from the Shopping mirror and compare PFP schemas across pages of the same functional type T. A model-assisted annotator extracts the schemas, and equivalent descriptions are merged before comparison. The analysis covers 6,238 pages; Table 9 reports the 6,229 pages in types with at least five examples. Their ag-

Table 8: Per-site scale of the world-model data. "Type change" counts records with $T' \neq T$, whereas "Within type" counts records with $T' = T$.

Site	Page types	Transitions	Type change	Within type
Shopping	45	615	501	114
CMS	176	2382	1880	502
GitLab	91	2395	1968	427
Reddit	27	374	284	90

gregate overlap of 0.891 supports the consistency of the shared inventory $\mathcal{I}(T)$ across the collected page instances.

Table 9: Within-type functional-primitive consistency on Shopping. Coverage is the mean overlap of reusable PFP schemas within each functional page type after equivalent template names are merged. The table reports page types with at least five pages.

Page type	Pages	Coverage
Category/listing	4534	0.889
Product detail	936	0.889
Search results	634	0.912
Shopping page	62	0.885
Account	17	0.875
Advanced search	17	0.945
Home	14	0.917
Cart	9	0.900
Order	6	0.858
Aggregate	6229	0.891

Cross-version functional stability. We compare 18 matched Amazon and Recreation.gov page states from their 2024 and 2026 production versions. PFP inventories are generated independently for each version, and snapshot-specific attributes are excluded before matching. Table 10 compares interactive elements, their structural locations, and functional primitives. PFPs retain 81.54% shared/union overlap, compared with 29.8% presence overlap for interactive elements. The 296 shared primitives among 326 historical and 333 current primitives yield 88.89% precision, 90.80% recall, and 89.83% F1. These matched snapshots show that page functions can remain stable as their concrete controls and layouts change.

Table 10: Cross-version representation stability on 18 matched production-site states. Agreement denotes presence overlap for interactive elements, structural-location overlap for interactive controls, and the shared/union ratio for functional primitives, of which 296 are shared.

Measure	Hist.	Curr.	Agreement
Interactive elements	–	–	29.8%
Structural locations	–	–	28.0%
Functional primitives	326	333	81.54%

C Few-Shot Cross-Site Generalization Details

Initialization and adaptation. We test whether WebArena dynamics training improves initialization for target-site transition prediction. Base and PFP-World use the same target-site support transitions and optimization settings; they differ only in initialization. K denotes labeled support transitions per site, and *Full* uses all support available in that panel.

Evaluation settings. Amazon uses a template-disjoint split with 45 test transitions. Its positive-support results micro-aggregate three training seeds after 10-epoch adaptation; $K{=}0$ reports the seed-42 direct-transfer diagnostic. The Practice Toolshop, DemoBlaze, and ExpandTesting Bookstore panel micro-pools 63 test transitions. Recreation.gov has seven test transitions and is reported separately as a change in both site and functional domain.

Metrics and matching. Transition-State F1 measures transition type, primitive additions and removals, and within-type updates, excluding primitives unchanged from the current inventory. Inventory Recall measures recovery of reference primitives in the successor inventory. Transition-Type Accuracy measures whether the prediction correctly identifies a type change or within-type transition. Inventory matching uses normalized canonical names on Amazon and conservative one-to-one semantic matching in the other panels. Additions and removals use exact multiset matching throughout; within-type updates use the fixed evaluator for each panel.

At $K{=}0$, Base outputs violate the required task-object schema. Daggers mark content-tolerant diagnostics that extract semantic content from those outputs. They provide a direct-transfer reference, while positive-support rows assess adaptation. Each result cell reports Base / PFP-WORLD; absolute values are compared within their respective panels.

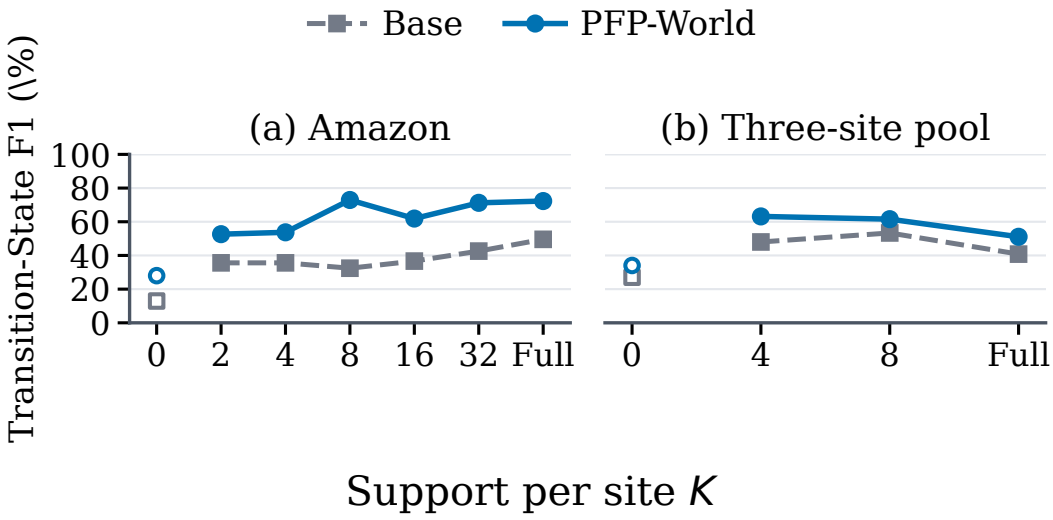


Figure 4: Few-shot Transition-State F1 on (a) template-disjoint Amazon and (b) three unseen shopping sites. Open $K{=}0$ markers denote direct-transfer diagnostics.

Support curves. Tables 11 and 12 report the complete curves. WebArena initialization improves Transition-State F1 at every positive-support budget in both aggregate panels. The per-site results in Table 13 show that this advantage is not uniform across individual sites: DemoBlaze favors Base at Full support, while the pooled result still favors PFP-WORLD.

Table 11: Complete Amazon support curve (%). Positive-support rows use fixed 10-epoch adaptation and pool predictions over three seeds. †Diagnostic $K{=}0$ Base output; see text.

K	Transition-State F1	Inventory Recall	Transition-Type Accuracy
0†	12.94 / 28.00	91.67 / 15.15	24.44 / 100.00
2	35.60 / 52.63	0.00 / 33.33	75.56 / 84.44
4	35.60 / 53.73	0.00 / 30.81	75.56 / 84.44
8	32.42 / 72.93	16.67 / 91.67	65.19 / 100.00
16	36.65 / 61.90	0.00 / 91.67	75.56 / 91.11
32	42.58 / 71.24	0.00 / 88.89	75.56 / 98.52
Full (56)	49.56 / 72.29	19.44 / 91.67	80.74 / 91.85

Table 12: Complete pooled support curve (%) for Practice Toolshop, DemoBlaze, and ExpandTesting Bookstore. *Full* uses 14, 12, and 11 support transitions, respectively. †Diagnostic $K{=}0$ Base output; see text.

K/site	Transition-State F1	Inventory Recall	Transition-Type Accuracy
0†	27.07 / 34.04	63.46 / 23.59	66.67 / 92.06
4	48.00 / 63.17	49.17 / 58.80	52.38 / 85.71
8	53.35 / 61.54	50.50 / 67.11	50.79 / 93.65
Full	40.79 / 51.08	35.88 / 60.47	53.97 / 92.06

Table 13: Per-site Transition-State F1 (%). Each cell reports Base / PFP-WORLD; *Full* uses the site-specific support count in Table 12. †Diagnostic $K{=}0$ Base output; see text.

K/site	Toolshop	DemoBlaze	Bookstore
0†	20.43 / 38.36	25.15 / 27.50	35.75 / 38.23
4	56.49 / 61.54	42.72 / 57.28	41.76 / 70.91
8	56.10 / 61.19	52.63 / 55.12	50.76 / 68.97
Full	40.92 / 45.63	55.52 / 50.79	13.26 / 58.68

Recreation.gov stress test. Recreation.gov changes both website and functional domain, but its test split contains only seven transitions. It provides a small mixed-result stress test and is not combined with either main panel.

Table 14: Recreation.gov few-shot state prediction (%). Each cell reports Base / PFP-WORLD. †Diagnostic $K=0$ Base output; see text.

K	Transition-State F1	Inventory Recall	Transition-Type Accuracy
0†	7.48 / 29.81	58.00 / 10.00	42.86 / 71.43
2	13.79 / 36.76	0.00 / 12.00	57.14 / 71.43
4	12.07 / 10.53	0.00 / 0.00	57.14 / 57.14
Full (8)	12.07 / 35.71	0.00 / 18.00	57.14 / 71.43

D IMPLEMENTATION DETAILS

This section specifies the shared module interfaces, retrieval index, and SFT input and output mappings for the planning variants.

STATE ABSTRACTOR prompt. The prompt asks the backbone to emit the current page's type-level inventory $\mathcal{I}(T)$ as a JSON list of strings, each describing a general PFP. It contains a role statement, the output schema with slot placeholders such as `{query}` or `{product}`, and the current page representation, either a DOM excerpt or a screenshot. Retrieval uses the unmodified checkpoint with the validated inventory index; SFT supervises abstraction with canonical inventories. Component evaluation ignores inventory order and requires the full canonical primitive set to match.

PRIMITIVE SELECTOR prompt. PRIMITIVE SELECTOR receives the user goal, current primitive inventory, and branch history. Each branch is initialized from the real execution history and subsequently carries its own predicted primitives and outcomes, including within-type updates. The selector ranks the inventory's PFPs by their contribution to the goal in this context, returning three root candidates and one continuation per branch at later depths. PFP slots remain unbound. Retrieval and SFT use the unmodified Qwen3-VL-8B-Instruct checkpoint for selection.

DYNAMIC SIMULATOR inputs and outputs. DYNAMIC SIMULATOR takes the current inventory and selected general PFP as its two inputs. It predicts the successor inventory, transition type, and any within-type state update. A change in functional page type determines the transition type independently of browser navigation. A within-type transition retains the inventory and describes its functional effect in the update, such as results being sorted, filters being applied, or the comparison set being updated. The one-step component metric matches predicted and reference inventory primitives semantically.

Branch history. After each prediction, the planning loop appends a record of the selected general PFP and its predicted effects, including any within-type update, to the branch history according to Eq. 2. The successor inventory is passed separately as the next rollout state. PRIMITIVE SELECTOR uses that inventory and the updated history, together with the user goal, to choose the next PFP.

TRAJECTORY SELECTION. TRAJECTORY SELECTION receives the user goal, execution history, and the candidate trajectories. At the default depth, these are three depth-3 sequences of general PFPs and predicted transition outcomes. The module compares their consequences, selects one trajectory, and binds its first PFP's slots to the required entities or values. Its output $\gamma^\star$ is this parameterized primitive as a whole. Slot values may be descriptive references, such as "the highest-rated product on the current page," which ACTION GROUNDING resolves against the browser observation. Later predicted steps support selection of the current plan and are not executed in advance.

ACTION GROUNDING. ACTION GROUNDING receives $\gamma^\star$, the current browser observation, and the user goal. It locates the relevant elements and generates the text-entry, click, or submit actions that realize the plan.

DUAL JUDGE and next-loop conditioning. After execution, the PFP-level judge assesses the selected primitive and its execution using the new observation and parameterized primitive. When the PFP remains appropriate but execution fails, ACTION GROUNDING retries the current plan. Other outcomes proceed to the task-level judge, which uses the new observation and user goal to assess task completion. A completed task ends the run. Otherwise, the actual outcome updates the execution history, and the new observation starts the next planning cycle.

Retrieval implementation. The world-model data are indexed by site and page-type identifier T; transition records are further indexed by the selected PFP τ. The retrieval adapter returns $\mathcal{I}(T)$ as the rollout state and uses (T, τ) to retrieve the recorded successor inventory, transition type, and any within-type update. For a type change, it returns the inventory of the resulting page type; for a within-type transition, it retains $\mathcal{I}(T)$ and returns the page-internal update. The page type T remains internal lookup metadata. An unseen (T, τ) pair falls back to the open-weight DYNAMIC SIMULATOR, and the miss is logged.

SFT implementation. The SFT data contain `state_abstractor` rows that map a page observation to its canonical inventory $\mathcal{I}(T)$, and `dynamics_simulator` rows for transition learning. Each dynamics row contains the pre-transition inventory, selected general PFP, successor inventory, transition type, and any within-type update. The model input contains the pre-transition inventory and selected PFP; the supervised output contains the successor inventory, transition type, and within-type update. Both modules use teacher-forced next-token log-likelihood. The data contain no benchmark goals, checkpoints, or task identifiers, and no SFT examples are constructed for PRIMITIVE SELECTOR.

Rollout-depth ablation. The depth ablation uses separate Retrieval and SFT configurations on the same sampled Shopping subset. At $d{=}0$, each retains its state abstractor and other model settings but selects a PFP directly without transition simulation. The selected primitive is parameterized and executed through the usual downstream modules. Both variants achieve 52.8%; positive-depth settings use the corresponding dynamics. Table 15 summarizes the shared settings and effects examined by the depth, Prompt, and initialization comparisons. Increasing depth also increases inference work, and the Prompt comparison uses the GPT-5.2 planning configuration described in Appendix F. The compared planning methods use their respective planning and feedback procedures.

Table 15: Experimental comparisons and shared settings.

Comparison	Shared setting	Effect examined
Direct PFP selection vs. depth-3 rollout (Table 4)	PFP representation, each variant's STATE ABSTRACTOR and remaining model settings, and the same Shopping subset	The additional value of predictive rollout.
Depth 1 vs. depth 3 (Table 4)	Dynamics variant, remaining model settings, and evaluation subset	The value of multi-step prediction over one-step prediction.
Prompt vs. PLAN–ACTION–REFLECT (Table 3)	Backbone, ACTION GROUNDING, and task evaluation	Functional-primitive planning without collected world-model data or parameter updates.
Base vs. WebArena initialization at positive support budgets (Appendix C)	Target-site support transitions, adaptation settings, and evaluation within each panel	Source dynamics training for target-site transition prediction.

Feedback ablation. We evaluate the feedback path using PFP-WORLD-SFT at depth 3 on all 108 CrossPageBench Shopping task instances. This experiment uses the same task set as the main Shopping evaluation. Table 16 shows a decline from 64.5% to 59.8% when execution checks and

next-loop feedback are removed together, a difference of 4.7 percentage points. The checks and feedback are coupled because the judges produce the outcome information used for feedback.

Table 16: Joint ablation of execution checks and next-loop feedback for PFP-WORLD-SFT at depth d=3 on the full CrossPageBench Shopping task set, reporting average target completion (%).

Variant	Feedback path	Completion
Full PFP-WORLD-SFT	on	64.5%
Without feedback path	off	59.8%

E WORLD-MODEL DECISION REVERSAL CASES

We inspect matched baseline and PFP-WORLD trajectories to examine how functional planning changes subsequent execution. This qualitative analysis complements the aggregate results. Concrete queries, products, and sorting criteria describe parameterized plans or execution outcomes; rollout candidates retain their general functional form. Cases spanning several stages summarize successive planning and execution cycles.

The successful cases illustrate how functional plans organize search, product comparison, and verification across planning cycles. The failures in Table 18 show selected plans that did not produce the changes required by the task.

Table 17: Successful decision-reversal cases. Each row summarizes a matched baseline/PFP-WORLD trajectory pair, its functional plans, and the observed outcome.

Case	Site	Task	Baseline behavior	PFP-WORLD behavior	Outcome
S1	Shopping	Coffee/blender	Used the header search box; the first action left the run in a malformed `coffee makerENTER` search state.	Selected Advanced Search, then executed a Product Name search parameterized with `coffee maker`, before continuing to comparison and checkout.	Reached the intended advanced-search result page.
S2	Shopping	Coffee/blender	Opened a stale Compare Products page immediately and kept revisiting compare/product pages.	Treated stale compare contents as a state hazard: clear unrelated items, add the coffee result, then return to Advanced Search for the blender branch.	Restored the intended coffee-to-blender sequence.
S3	Shopping	Coffee/blender	Repeated local repair on the Keurig product page and missed the second product.	Selected the highest-rated blender add-to-compare action, then moved from compare to cart and checkout.	Compare count reached two and the run entered checkout shipping.
G1	GitLab	G-11-H1	Reached the Starred projects page but scrolled instead of switching the sort order.	Selected the sorting function after comparing search and view-starred candidates, with Name as the execution criterion.	Task verifier returned `task_done`.
G2	GitLab	G-13-H3	Did not close the final Internal-list verification for a multi-file snippet.	The run followed the required sequence of title, visibility, file fields, comments, visibility change, and Internal-list verification.	Completed the snippet task and verified Internal visibility.

Table 18: Failure cases from the decision-reversal analysis, showing selected plans and the task requirements that remained unmet during execution.

Case	Failure summary
F1 Shopping	The selected trajectory placed wishlist removal after ordering. Its first executed plan was Place Order, but the run produced neither order-confirmation nor wishlist-removal evidence. This case shows how a plausible later continuation can fail to make the first step useful for completing the task.
F2 GitLab	After checkbox trouble, the selected plan navigated through User Settings instead of completing the checkbox and Save operations. The broader navigation did not produce the state change required by the task.

F REPRODUCIBILITY

Models and implementation variants. Retrieval and SFT use Qwen3-VL-8B-Instruct for the front-end planning models. Retrieval combines an unmodified state abstractor with the validated page-type index and retrieves recorded transitions; unseen type–PFP pairs fall back to the open-weight simulator. SFT updates STATE ABSTRACTOR and DYNAMIC SIMULATOR, while PRIMITIVE SELECTOR uses the unmodified checkpoint. TRAJECTORY SELECTION, ACTION GROUNDING, and DUAL JUDGE use `gpt-5.2` in both variants. Prompt uses `gpt-5.2` for these modules and all three front-end planning modules, without collected world-model data or parameter updates.

The reproduced WMA and RLVR-World evaluations use `gpt-5.2` as the policy backbone. PLAN–ACTION–REFLECT and AWM also use `gpt-5.2`; the former shares our grounding module while replacing upstream planning, and the latter retains its workflow-memory design.

Training and planning. SFT learns observation-to-inventory and inventory–PFP-to-transition mappings with teacher-forced next-token log-likelihood. Appendix D describes the input and output mappings. Target-site adaptation compares base and WebArena-trained initializations under matched support data and optimization settings, with panel-specific details in Appendix C. The default planner uses depth 3, three root candidates, and one continuation per branch at later steps.

Hardware. Open-weight inference and fine-tuning run on a host with 8×NVIDIA RTX 4090 GPUs. Multi-GPU runs set `NCCL_P2P_DISABLE=1 NCCL_IB_DISABLE=1` to avoid NCCL deadlocks observed on this hardware.

Browser execution and stopping. WebArena and CrossPageBench use the same headless browser runner with a per-run execution timeout. Runs that reach the timeout remain in the evaluation: they count as failures on WebArena, while CrossPageBench retains partial credit for achieved checkpoints.

Repeated runs and reporting. Each WebArena and CrossPageBench task is executed three times, and its highest score across the three runs is used for reporting. WebArena reports end-to-end success rate, while CrossPageBench reports average target completion. The rollout-depth ablation uses a sampled Shopping subset and reports separate Retrieval and SFT baselines at depth zero, each retaining its state abstractor and remaining model configuration; the feedback ablation uses all 108 Shopping instances with SFT at depth 3. Model usage and runtime are reported per matched control-loop iteration, including planning, grounding, browser execution, and judging.

F.1 MODEL USAGE AND RUNTIME

Table 19 compares WMA and PFP-WORLD-SFT over a complete control-loop iteration, including WORLD MODEL PLANNING, ACTION GROUNDING, browser execution, and DUAL JUDGE. It reports mean model calls, tokens, and client-observed runtime on matched runs. We report direct usage rather than monetary cost because the configurations combine local and API models.

Relative to WMA, depth 1 uses 79.9% fewer model calls and 74.1% fewer tokens per iteration; depth 3 reduces them by 59.1% and 64.6%. Runtime is longer despite this lower token usage. PFP-

Table 19: Mean model calls, total tokens (input plus output), and client-observed runtime per matched control-loop iteration; PFP-WORLD uses the SFT route.

Method	Calls	Total tokens	Time (s)
WMA	24.914	105,591.1	40.96
PFP-WORLD-SFT (d=1)	5.008	27,380.4	48.96
PFP-WORLD-SFT (d=3)	10.179	37,418.9	97.66

WORLD-SFT runs planning, grounding, and feedback sequentially, and depth 3 also incurs model-loading and device-placement overhead. The measurements describe resource use per control-loop iteration. Because a PFP iteration may cover several atomic actions, task-level compute also depends on the number of iterations required.

F.2 END-TO-END CROSS-SITE EVALUATION

We evaluate PFP-WORLD-SFT and PLAN–ACTION–REFLECT on 20 user-scenario tasks, with 10 each on Amazon and Practice Toolshop. The tasks include selecting products, adding items to the cart, and editing account information. We use the average target completion metric defined in Sec. 5.1, taking the highest score over three executions of each task before averaging across tasks.

For each target site, we adapt the WebArena-trained DYNAMIC SIMULATOR using the Full support set in Appendix C: 56 transitions for Amazon and 14 for Practice Toolshop. PFP-WORLD-SFT uses rollout depth d=1. STATE ABSTRACTOR and PRIMITIVE SELECTOR retain the Qwen3-VL-8B-Instruct configurations of the main SFT experiments; TRAJECTORY SELECTION, ACTION GROUNDING, and DUAL JUDGE use `gpt-5.2`. PLAN–ACTION–REFLECT retains the `gpt-5.2` configuration used in the main comparison. Table 20 reports the aggregate results.

Table 20: Average target completion (%) over 20 user tasks, with 10 each on Amazon and Practice Toolshop. PFP-WORLD-SFT uses dynamics adapted with each target site's Full support set.

Method	Average target completion (%)
PLAN–ACTION–REFLECT	71.4
PFP-WORLD-SFT	**75.3**