Diagnosing Semantic Handoff Failures in Agent-Orchestrated Vision-Language-Action Skill Composition

Ke Rui*, Yushen Zuo*, Jiawei Wang*†, Haoran Jia, Jinming Ma, Weitao Zhou, Minglei Li†
SimpleAI
*Equal contribution †Corresponding authors: {wangjiawei, liminglei}@simpleai.tech

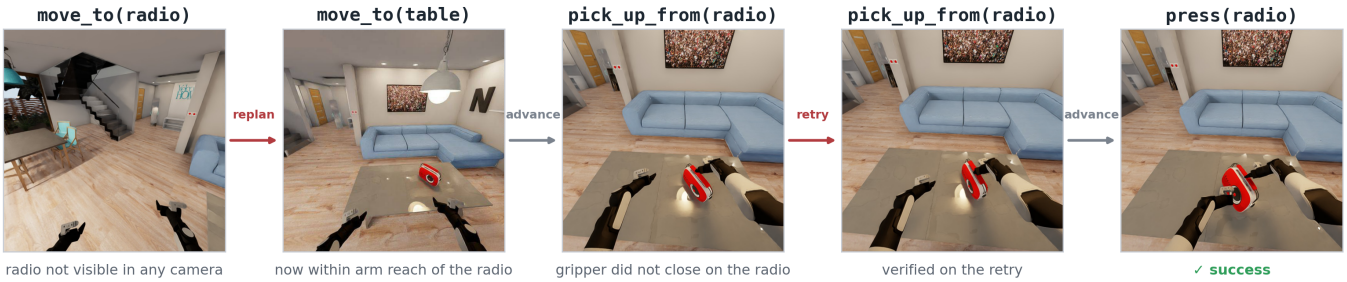

Fig. 1: **Turning on the radio through skill composition.** The agent issues a sequence of typed skill calls such as `move_to`, `pick_up_from`, and `press`, and a multi-view VLM verifier checks each handoff before the next skill runs. When a check fails the agent recovers within the same loop—re-planning to a new sub-goal after a navigation that does not reach the radio, and retrying after a grasp that does not close—advancing only once the post-condition holds, until the radio turns on. This plan-act-verify-replan loop is our agent harness; Fig. 2 gives the full architecture.

***Abstract*—Long-horizon household tasks require robots to compose many language-conditioned skills, but the boundary between two skills is rarely explicit. A skill may satisfy its own postcondition while leaving the robot, objects, or camera views in a state from which the next skill cannot start. We study this *semantic handoff* problem in BEHAVIOR-1K through an agent-orchestrated vision-language-action execution harness. The harness calls $\pi_{0.5}$-based skill checkpoints trained from cleaned BEHAVIOR-1K demonstrations, assigns each skill typed arguments and a step budget, and uses multi-view VLM verification to decide whether execution should advance, retry, or replan. To separate isolated skill competence from long-horizon composition, we compare the same checkpoints from clean skill-boundary snapshots and from chained terminal states produced by previous skills. Selected navigation, grasping, placement, and door-opening skills reach 77–100% success from snapshots under human-reviewed verification, yet composed rollouts still stall from chained states. The resulting traces attribute failures to next-skill readiness, target grounding, and control execution, turning near-zero task success into actionable diagnostics for what VLA skill libraries must learn next: robustness to the messy chained-state distribution that clean demonstrations underrepresent.**

I. Introduction

Long-horizon household tasks require robots to maintain semantic state across many visually grounded actions. Making microwave popcorn, for example, means navigating to the kitchen, opening the microwave, inserting the bag, closing the door, and activating the appliance. No single language-conditioned action solves such a task; it requires composing many skills while preserving the conditions that make the next skill executable.

A natural architecture is to let an agent orchestrate vision-language-action (VLA) skills. In this view, VLA policies act as general-purpose, language-conditioned visuomotor tools, while the agent layer handles task decomposition, state tracking, verification, recovery, and evidence collection. Compared with hand-engineered task and motion planners or symbolic tool APIs[20, 8], this boundary promises broader coverage: new objects and appliances need not each come with a new symbolic controller. The challenge is that learned skills expose a less explicit interface. They may complete the local behavior requested by the agent without producing a state that is usable by the next skill.

We call this the *semantic handoff* problem (Sec. II). A skill can satisfy its own postcondition yet leave the robot, objects, or camera views in a state from which the following skill cannot start. For example, a navigation skill before a grasp should not be considered complete merely because the target object is visible somewhere in the scene; the object should be close enough and positioned plausibly for grasping. Similarly, opening a door may satisfy an open-door postcondition while moving the next target out of view, creating a downstream grounding failure. Thus, long-horizon execution requires judging not only whether the current skill appears complete, but whether the resulting state is ready for the next skill.

We study this problem in BEHAVIOR-1K [17] using a com-

pact $\pi_{0.5}$-based VLA skill library trained from cleaned skill demonstrations and an agent execution harness that performs bounded plan-act-verify-replan control. The harness represents each skill call with typed arguments, a step budget, and a handoff-aware postcondition; executes the corresponding VLA checkpoint; verifies progress from head and wrist observations with a multi-view VLM verifier; and records replayable traces of skill attempts, verifier decisions, replans, and final task predicates. Our goal is not to claim state-of-the-art BEHAVIOR-1K task success. Instead, we ask what such an agent harness can reveal when long-horizon rollouts fail.

This diagnostic framing separates isolated skill competence from compositional robustness. We evaluate the same skill checkpoints both from clean skill-boundary snapshots and from chained terminal states produced by previous skills. Several skills succeed frequently from curated snapshots under human-reviewed verification, yet composed rollouts still stall from the messy states produced during execution. The resulting traces turn near-zero task success into actionable evidence: failures can be attributed to next-skill readiness, target grounding, or control execution from chained states, pointing to the data and verification capabilities that VLA skill libraries must improve.

This paper makes three contributions:

- We formulate long-horizon agent-orchestrated VLA execution as semantic skill composition, where each skill call has typed arguments, a budget, an expected postcondition, and an implicit next-skill readiness requirement; we identify semantic handoff failures as cases where a skill appears complete but does not leave the state ready for the next skill.
- We instantiate this formulation as a BEHAVIOR-1K semantic execution harness: $\pi_{0.5}$-based skill checkpoints trained from cleaned skill segments are invoked through a bounded plan-act-verify-replan loop with multi-view VLM verification, handoff-aware postconditions, recovery, and replayable trace artifacts.
- We provide trace-backed diagnostics showing that snapshot-state skill competence does not by itself carry over to chained initial conditions. On a representative BEHAVIOR-1K rollout round, the harness attributes failed skill attempts to next-skill readiness, target grounding, and control execution from chained states; a controlled arm-reach verifier ablation surfaces additional readiness failures and triggers more re-navigation attempts.

II. Semantic Handoff Problem

Long-horizon execution chains language-conditioned skills, but the boundary between two skills is under-specified. Each skill is trained and benchmarked to reach its own postcondition ϕ_t from a curated skill-boundary state; in composition, however, the next skill s_{t+1} must start from whatever state the previous skill s_t actually leaves behind. A skill can therefore satisfy ϕ_t and still strand the robot, objects, or camera views in a state from which s_{t+1} cannot execute—a *semantic handoff* failure. The agent must judge not merely whether the current state is a plausible completion of s_t, but whether it is a valid *starting* state for s_{t+1}.

A standard verifier checks only ϕ_t, which is insufficient at a boundary. For instance, a `move_to(can)` step followed by `pick_up_from(can)` should not advance merely because the can is visible at a distance; it should advance only when the can is also plausibly reachable by the gripper. The same checkpoint is also exercised on different inputs in the two settings: a single-skill benchmark evaluates `move_to` from a restored demonstration snapshot, whereas an end-to-end rollout invokes it after the robot has navigated, opened doors, and shifted viewpoints, so the next skill no longer starts from the curated skill-boundary distribution. This competence drop at skill boundaries is a distribution shift in the imitation-learning sense [24], but distinct from classic within-rollout compounding error: each skill's legitimate terminal state is already out of distribution for the *next* skill—a shift between two composed policies rather than within a single one.

In the current system we approximate this handoff requirement directly in the skill postconditions rather than through a separate predicate: the natural-language postcondition authored for each `move_to` step requires the target to be shown close-up and at arm-reach distance, not merely visible. This is a deliberately limited instantiation; a principled, next-skill-indexed readiness predicate $\rho(s_t, s_{t+1})$ verified jointly with the postcondition, $\phi_t \wedge \rho$, is a design direction we develop in Sec. V-C.

III. Semantic Execution Harness

Our harness separates the agent layer from the embodiment layer. The agent layer selects skill calls, maintains task state, invokes verification, and triggers recovery. The embodiment layer owns simulation, VLA inference, and low-level action execution. Figure 2 summarizes the system.

A. Semantic Skill Contracts

We represent a skill call as

$$s_t = (\text{name}, \text{args}, \text{prompt}, B_s, K_s, \phi_t),$$

where name selects a skill type, args bind task objects, prompt is the language instruction passed to the VLA policy, B_s is the maximum skill budget, K_s is the verifier interval, and ϕ_t is the expected postcondition. The current skill library includes `move_to`, `pick_up_from`, `place_on`, `place_on_next_to`, `place_in`, `press`, `turn_on_switch`, `open_door`, and `close_door`. We use these names throughout; trace artifacts label `move_to` and `pick_up_from` as `navigate_to` and `grasp`. `press` and `turn_on_switch` are distinct skills that share one checkpoint group: a button/switch press versus a toggle-on action.

The VLA executor is trained as part of the system rather than assumed as an external oracle: the skills are $\pi_{0.5}$ vision-language-action policies [3], a PaliGemma vision-language backbone with a flow-matching action expert over three RGB views from the head and two wrists, plus proprioceptive state.

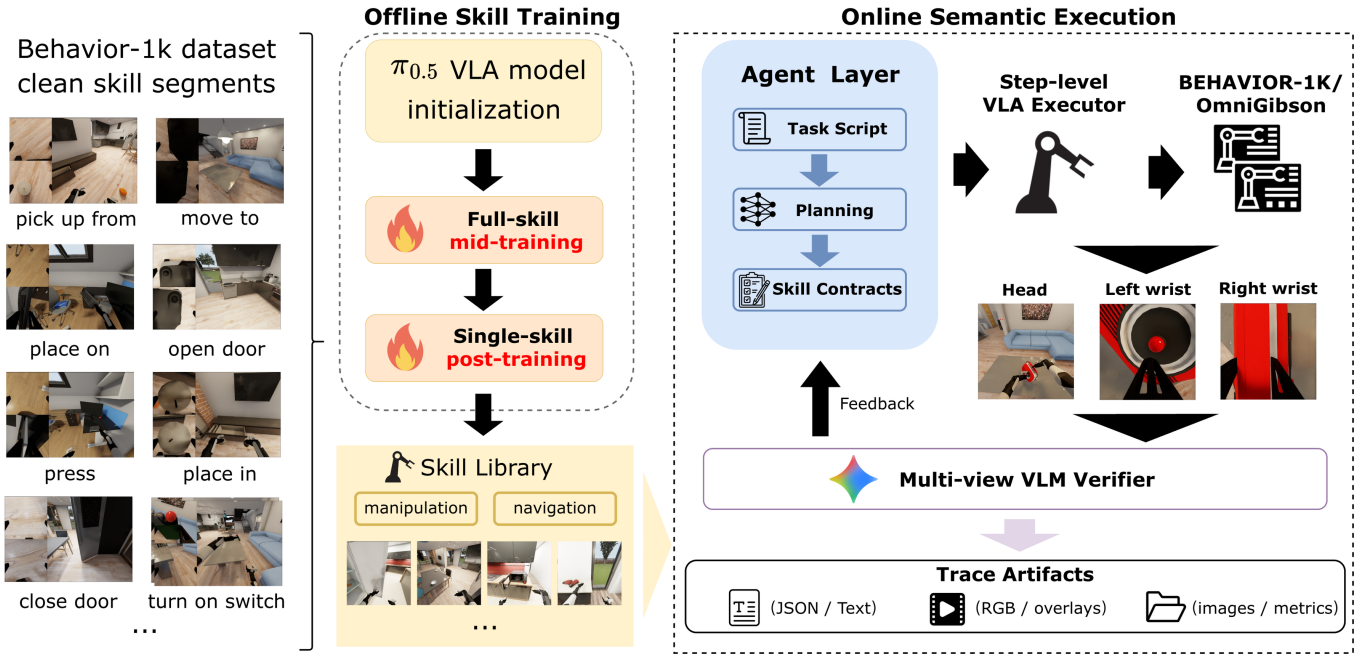

Fig. 2: **The semantic execution harness.** *Offline* (left), cleaned BEHAVIOR-1K demonstrations initialize a $\pi_{0.5}$ VLA backbone that full-skill mid-training and single-skill post-training specialize into a skill library spanning manipulation and navigation. *Online* (right), the agent layer maintains a task script, planning, and typed skill contracts, and dispatches step-level VLA skills in BEHAVIOR-1K/OmniGibson; a multi-view VLM verifier judges progress from head and wrist views and feeds its verdict back to the agent to advance, retry, or replan. Every rollout writes replayable trace artifacts (logs, videos, and evidence).

We clean BEHAVIOR-1K teleoperation demonstrations [17] into skill segments and train in two stages: a pooled all-skill mid-training, then skill-group specialization, with each skill type routed to its group's checkpoint at deployment (Table V). We use skill-specific budgets B_s because VLA policies lack a reliable stop signal and BEHAVIOR-1K skill durations vary widely (Table IV).

B. Step-Level Plan-Act-Verify-Replan

For each skill, the harness executes a VLA action chunk for K_s environment steps, collects head and wrist observations, and queries a VLM verifier over the head, left-wrist, and right-wrist views together with the current skill's postcondition. The skill advances as soon as a verification returns a positive verdict above a confidence threshold of 0.6; an `uncertain` verdict never terminates a skill, a fail-safe against flaky calls. Otherwise execution continues until the skill budget is exhausted or the agent replans. Because the agent, not the VLA, owns termination, every rollout produces a step-resolved verifier trajectory rather than a single end-of-skill verdict.

IV. Diagnostic Evaluation

A. Setup

We evaluate in BEHAVIOR-1K with two complementary protocols: an isolated *single-skill* benchmark and a composed *end-to-end* rollout. The single-skill benchmark restores the simulator snapshot at a cleaned demonstration segment's start frame from the validation split and runs the corresponding skill-group checkpoint in isolation, scoring success with the multi-view VLM verifier whose decisions are confirmed by human review. These snapshots are drawn from the same task instances we evaluate end-to-end, so the isolated-vs-composed comparison holds the scene and instance fixed and varies only the skill's initial-state distribution—a clean snapshot versus the chained terminal state of the previous skill. The end-to-end protocol composes the same checkpoints through the agent loop under live VLA execution, each skill starting from the terminal state of the previous one; we run official BEHAVIOR-1K public test instances per task, taken from the 2025 challenge `test_instances.csv`, one instance per round across three rounds. Execution is step-level plan-act-verify-replan: the agent verifies every $K = 200$ simulator steps with `gemini-2.5-flash` multi-view verification and advances a skill only when its postcondition holds, including an arm-reach handoff criterion on navigation, terminating on a per-skill step budget and a global episode budget of twice the human-demonstration length. Every rollout writes a structured trace of skill dispatches, verifier decisions, replans, final task-predicate checks, and media, which we parse into per-skill attempts; failed attempts are classified from structured verifier reasons as diagnostic labels, not human ground truth. The progress score aggregates all three rounds (n=3); the per-attempt failure attribution and verifier-call diagnostics are reported on one representative round.

Skill	Success	Total	SR
move_to	27	35	77.1%
pick_up_from	28	29	96.5%
place_in	11	13	84.6%
place_on	6	6	100.0%
open_door	7	7	100.0%
close_door	4	6	66.7%
press	–	–	–
turn_on_switch	7	8	87.5%

TABLE I: **Isolated VLA skill benchmark results from BEHAVIOR-1K skill instances.** Long-horizon deployment additionally requires robustness to chained initial conditions. Both protocols use the same skill-group checkpoints; the isolated benchmark serves them through the single-skill evaluation path, while the end-to-end harness routes them through the agent loop. Success is scored by the multi-view VLM verifier with human review of its decisions, rather than the simulator BDDL predicate, on validation-split snapshots drawn from the same instances evaluated end-to-end. Per-skill samples are small, so exact rates should not be over-interpreted; `press` shares the switch/press checkpoint with `turn_on_switch` and was not separately benchmarked in isolation.

B. Several Skills Are Competent From Curated Snapshot States

Table I summarizes the isolated skill benchmark. Several skills achieve strong judge success from curated skill-boundary states: `place_on` reaches 6/6, `open_door` reaches 7/7, `pick_up_from` reaches 28/29, `place_in` reaches 11/13, and `move_to` reaches 27/35. Notably, the placement skills—which dominate the end-to-end bottlenecks below—are themselves competent from snapshot states once trained to convergence, as Table V shows, so their chained failures cannot be attributed to weak placement policies in isolation. The skill library is therefore not uniformly weak in isolation.

C. Competence Does Not Survive Composition

The single-skill and end-to-end protocols evaluate different semantic states. In the single-skill benchmark, a grasp skill starts with the object, robot pose, and camera view close to a successful demonstration boundary. In end-to-end execution, skill s_t starts from the actual terminal state produced by s_{t-1}: shifted gripper poses, displaced objects, off-angle camera views, or ambiguous object instances. The result is a skill-to-task gap: VLA skills that are competent from snapshot states become unreliable when called from chained states. Crucially, both protocols route the *same* skill-group checkpoints (Table I): although they use different success criteria—isolated skill success is VLM-judged with human review, composed task success is the BDDL predicate—the checkpoint that scores highly in isolation is exactly the policy that stalls in composition. The gap is not visible in isolated snapshot evaluation alone: because the same checkpoints are used in both protocols, the traces suggest that chained boundary states expose a robustness failure that is not explained by isolated skill success rates. We hypothesize that simply scaling the same snapshot-only training distribution would not fully close this gap; such scaling adds density on the curated boundary distribution, where the skills are already competent, rather than coverage of the chained states where the observed failures concentrate. Targeted chained-state data and readiness-aware verification are therefore likely necessary.

Task	Ref.	Progress (%)
Turn on radio	4	50.0 ± 43.3
Make microwave popcorn	8	45.8 ± 7.2
Move boxes to storage	10	30.0 ± 0.0
Pick up trash	12	22.2 ± 4.8
Put shoes on rack	14	11.9 ± 4.1
Bring in wood	13	10.3 ± 4.4
Cook hot dogs	13	10.3 ± 4.4
Set mousetraps	12	8.3 ± 0.0
Freeze pies	26	6.4 ± 2.2
Hide Easter eggs	12	0.0 ± 0.0

TABLE II: **Per-task progress score across the ten BEHAVIOR-1K tasks, three instances each.** The progress score is the fraction of the task's reference skill sequence (**Ref.**, the BEHAVIOR-1K teacher-hint plan) that a rollout completes. We score it by human inspection of the rollout video against the reference skill sequence: a step is counted complete only when the video shows its postcondition satisfied. Because alternative valid plans are not credited and scoring is a single annotator's judgement, this is a conservative diagnostic proxy rather than an official BEHAVIOR-1K metric. We report the mean ± standard deviation over the three instances (n=3). It recovers a graded signal where task-level predicate success is near zero: a rollout can clear most of its reference plan yet still miss the final task predicate (e.g. a completed radio sequence still leaves the appliance not registered as on). Progress concentrates in early navigation, door, and grasp steps and collapses at contact-rich manipulation and late multi-object placement.

D. Attributing Handoff Failures to Actionable Causes

Table II reports a quantitative per-task signal over three instances of each task: a *progress score* equal to the fraction of the task's reference skill sequence, the BEHAVIOR-1K teacher-hint plan, that a rollout completes, scored by hand from the rollout video. Because task-level predicate success is near zero, this graded score is what separates a run that stalls at its first navigation from one that clears most of its plan: mean progress across the thirty rollouts is 19.5%, ranging from roughly half the reference plan on the radio and popcorn tasks down to zero on Easter eggs, and the score localizes where competence collapses—contact-rich manipulation and late multi-object steps.

Beyond this aggregate progress score, the harness attributes every failed attempt to a specific cause. The three trace-derived causes—target grounding or scene search, control execution,

and next-skill readiness (Table III)—are all VLA-side; each points at a concrete skill and chained-state distribution to improve and maps to a targeted remedy. Crucially, all three are *transfer* failures: the same checkpoints are competent in isolation (Table I), so the breakdown cannot be diagnosed from isolated skill success alone. This is the payoff of the agent layer: it turns near-zero task *success* into a prioritized list of what the VLA must learn next. Notably, the binding failure at most handoffs is not premature advance through a weak state but the policy failing to *reach* a ready state from its chained start, exhausting the skill budget. Tightening the navigation verifier to require an arm-reach rather than a merely same-region handoff is a controlled ablation that changes only the verifier criterion, holding tasks, instances, and checkpoints fixed. This sharpens the readiness signal: it surfaces 12 additional readiness failures and 25 more re-navigation attempts and recovers the radio task, while leaving control/commit the dominant cause (Table VI). A blinded human audit of the verifier (Appendix, Table VII) confirms its failure decisions: of the verifier-flagged failures the annotators could adjudicate, 20 of 21 were judged real (over-strict 0.05), so a reported failure is a real failure rather than a verifier artifact. Because the failure *categories* remain verifier-derived and the audited set is small, we report the category counts as preliminary indicators rather than calibrated rates. Table III aggregates the failed-attempt categories; per-task attempt, verifier-call, and termination diagnostics appear in the appendix, drawn from a representative ten-rollout round, one instance per task, that issues 196 verifier calls in total.

V. Discussion

A. Implications

The results support reading the agent layer as a diagnostic instrument for long-horizon robotics, and should be read as preliminary diagnostics. The harness executes, verifies, and records each rollout, turning a budget-limited run into a step-resolved trace of where execution stalls. These traces reveal concrete blocked states where the remaining issue is most often the chained-state distribution the VLA was never trained on—a skill-boundary distribution shift—rather than a uniformly weak policy; target grounding is the other recurring factor. Further ablations that rerun the same task suite while varying handoff checks, verifier cadence, or recovery strength would measure whether the bottleneck moves; we leave these to future work.

B. Limitations

The current end-to-end evidence covers a small number of completed pilot traces, so bottleneck counts should not be read as population-level estimates. The verifier is VLM-based; a blinded two-annotator audit (Table VII) confirms its failure decisions (20 of 21 verifier-flagged failures judged real, over-strict 0.05), but it validates only the binary failure decision—the failure-category labels remain verifier-derived and may carry bias for object placement, grasp visibility, and articulated-object state, and the audited set is small. The task scripts use some under-specified object names such as `door` or `storage box`; robust instance-level binding remains open. Finally, recovery is handled by the agent replanning within the episode budget—in practice it usually re-navigates to re-establish the handoff before retrying a failed skill. A no-recovery versus replanning ablation, runnable in our harness, is designed to separate VLA skill brittleness from recovery limitations; an oracle skill-boundary reset remains a future upper bound because our current live execution setup does not expose arbitrary skill-start state restoration.

C. Future Work: Next-Skill Readiness Verification

The handoff failures diagnosed above motivate a principled treatment of next-skill readiness (Sec. II). Rather than hand-authoring an arm-reach clause into each `move_to` postcondition, the verifier could check

$$\phi_t \wedge \rho(s_t, s_{t+1}),$$

where $\rho(s_t, s_{t+1})$ is a next-skill readiness predicate generated from the type and arguments of s_{t+1}:

$$\rho(s_t, s_{t+1}) = R_{\text{name}(s_{t+1})}(\text{args}(s_{t+1}), o_t),$$

o_t is the current multi-view visual evidence and R is a finite template library indexed by the next skill type. For example, a `move_to` before `pick_up_from` would pass only when the target is plausibly graspable, whereas one before `open_door` would instead require the handle within reach (Table IX). Evaluating this design—in particular a postcondition-only versus handoff-aware ablation that isolates the effect of ρ—requires a verifier conditioned on the next skill and is left to future work.

VI. Related Work

Robot policy learning. Language-conditioned manipulation has progressed from modular imitation-learning systems such as CLIPort and PerAct [27, 28] and prompt-conditioned agents such as VIMA [14] to large robot policies and VLA models including Gato, RT-1, RT-2, Open X-Embodiment, Octo, OpenVLA, π_0, and diffusion policies [23, 4, 5, 22, 21, 16, 3, 7]. Recent work further strengthens the VLA substrate through world models and learned memory [6, 25]. We build a compact $\pi_{0.5}$-based VLA skill library from cleaned BEHAVIOR-1K skill data, but focus on the execution problem around that library: how trained skills should be composed, verified, recovered, and diagnosed under chained initial conditions.

Task and motion planning and programmatic policies. Task and motion planning combines symbolic task structure with continuous motion, grasp, placement, and collision constraints [15, 9]; code-generating approaches such as Code as Policies synthesize executable robot programs over perception and control APIs [20]. MAESTRO and TIGeR similarly enrich VLM agents with active perception, geometry, planning, control, or exact geometric-computation tools [26, 10]. H-WM guides task and motion planning with a hierarchical world model that curbs error accumulation across long-horizon VLA execution [11]. These systems show the value of stable tools

Attributed Cause	Sub-Category	n	Dominant Failure Signature	Targeted Remedy
Control execution	grasp	31	reaches the object but the grasp does not close or hold	contact-rich, commit-frame data
	actuation	15	reaches the handle, switch, or button but it does not actuate	
	placement	12	approaches the surface or container but mis-positions the object	
Target grounding	scene search	37	target out of view or the wrong instance after navigation or door opening	instance-grounding supervision
Next-skill readiness	navigation-to-ready	35	navigation stops short of an arm-reach pose at the handoff	chained-state fine-tuning

TABLE III: **Failed skill attempts attributed to three VLA-side causes, with sub-category breakdown and remedy** (live VLA, K=200, a representative round, one official instance per task; 130 failed attempts). All three are *transfer* failures of in-isolation-competent skills (Table I), not raw incapacity. Verifier-side cases (the two excluded attempts) are not counted here; counts derive from structured verifier reasons and are diagnostics, not ground-truth labels.

with inspectable semantics. Our harness treats trained VLA skills as agent-callable tools and supplies a missing runtime interface: typed skill contracts, handoff-sensitive postcondition checks, and trace-backed failure attribution.

Language-conditioned agents and semantic grounding. SayCan, Inner Monologue, and ReAct-style systems use language models to select, reason over, or call robot tools [1, 12, 31]. Recent agentic robot systems add structured workflows, multi-agent decomposition, temporal verification, unified data-collection/deployment loops, and stronger embodied VLM backbones [29, 30, 19, 2]. Semantic grounding work also uses language or vision-language models to build spatial constraints, value maps, or reusable atomic skills [13, 18]. We target execution-time grounding at skill handoffs: a step should advance not when the target is merely visible but when the state is ready for the next skill—a readiness notion we fold into the skill postcondition and check online rather than as an offline label.

VII. Conclusion

We presented a semantic execution harness for agent-orchestrated VLA long-horizon skill composition. The harness represents skills through typed semantic contracts, executes VLA policies under bounded step-level verification, and surfaces semantic handoff failures at skill boundaries. Preliminary BEHAVIOR-1K diagnostics show a gap between snapshot skill competence and chained-state robustness. This gap is best interpreted as a roadmap for VLA improvement: the agent layer supplies task control, verification, recovery, and evidence, while trace-backed semantic diagnostics identify the VLA recovery states and grounding capabilities needed for reliable long-horizon robots.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do As I Can, Not As I Say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[2] BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Xiansheng Chen, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, et al. RoboBrain 2.0 technical report. *arXiv preprint arXiv:2507.02029*, 2025.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[7] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[8] Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

[9] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International*

Conference on Automated Planning and Scheduling, 2020.

[10] Yi Han, Enshen Zhou, Shanyu Rong, Jingkun An, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Lu Sheng, and Shanghang Zhang. TIGeR: Tool-integrated geometric reasoning in vision-language models for robotics. *arXiv preprint arXiv:2510.07181*, 2025.

[11] Jinbang Huang, Wenyuan Chen, Zhiyuan Li, Oscar Pang, Xiao Hu, Lingfeng Zhang, Yuanzhao Hu, Zhanguang Zhang, Mark Coates, Tongtong Cao, et al. H-WM: Robotic task and motion planning guided by hierarchical world model. *arXiv preprint arXiv:2602.11291*, 2026.

[12] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner Monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, 2022.

[13] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. In *Conference on Robot Learning*, 2023.

[14] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.

[15] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011.

[16] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[17] Chengshu Li, Ruohan Zhang, Josiah Wong, et al. BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, 2023.

[18] Dongjiang Li, Bo Peng, Chang Li, Ning Qiao, Qi Zheng, Lei Sun, Yusen Qin, Bangguo Li, Yifeng Luan, Bo Wu, et al. An atomic skill library construction method for data-efficient embodied manipulation. *arXiv preprint arXiv:2501.15068*, 2025.

[19] Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. RoboClaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026.

[20] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation*, 2023.

[21] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[22] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023.

[23] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

[24] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[25] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[26] Junyao Shi, Rujia Yang, Kaitian Chao, Selina Bingqing Wan, Yifei Shao, Jiahui Lei, Jianing Qian, Long Le, Pratik Chaudhari, Kostas Daniilidis, et al. MAESTRO: Orchestrating robotics modules with vision-language models for zero-shot generalist robots. *arXiv preprint arXiv:2511.00917*, 2025.

[27] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. CLIPort: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, 2022.

[28] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2022.

[29] Yi Yang, Kefan Gu, Yuqing Wen, Hebei Li, Yucheng Zhao, Tiancai Wang, and Xudong Liu. ManiAgent: An agentic framework for general robotic manipulation. *arXiv preprint arXiv:2510.11660*, 2025.

[30] Zhejian Yang, Yongchao Chen, Xueyang Zhou, Jiangyue Yan, Dingjie Song, Yinuo Liu, Yuting Li, Yu Zhang, Pan Zhou, Hechang Chen, et al. Agentic robot: A brain-inspired framework for vision-language-action models in embodied agents. *arXiv preprint arXiv:2505.23450*, 2025.

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

APPENDIX

Training and Routing Details

The skills are $\pi_{0.5}$ vision-language-action policies [3]: a PaliGemma 2B vision-language backbone coupled with a 300M-parameter flow-matching action expert that consumes three 224×224 RGB views from an egocentric head camera and two wrist cameras together with proprioceptive state and predicts multi-step action chunks. We clean BEHAVIOR-1K teleoperation demonstrations [17] into skill segments aligned with the skill contracts using the dataset's fine-grained sub-task annotations, with a per-task split of 180 training and 20 validation episodes. Training has two stages. A full-skill mid-training stage finetunes the base $\pi_{0.5}$ checkpoint over the pooled skill data for 50k steps to preserve shared visuomotor and language grounding. From this initialization, skill-group post-training specializes separate checkpoints—navigation/orientation, switch/press, door open/close, grasping, surface placement, and containment placement—with full-parameter updates under a cosine learning-rate schedule (peak 2.5×10^{-5}, global batch size 256). At deployment the agent routes each skill type to its group's checkpoint and renders it into the skill's language prompt (Table V).

Skill	Mean Frames	Mean Seconds
`move_to`	692	23.1
`pick_up_from`	598	19.9
`place_on`	430	14.3
`place_in`	346	11.5
`press`	291	9.7
`open_door`	941	31.4
`close_door`	505	16.8

TABLE IV: **Skill-duration statistics from selected BEHAVIOR-1K demonstrations.** The spread motivates skill-specific budgets and bounded verification.

Skill	Checkpoint Group	Steps	Budget B_s
`move_to`	nav./orient.	60k	800
`press`	switch/press	60k	800
`turn_on_switch`	switch/press	60k	800
`open_door`	door open/close	60k	1500
`close_door`	door open/close	60k	800
`pick_up_from`	grasping	180k	800
`place_on`	surface placement	180k	800
`place_on_next_to`	surface placement	180k	1200
`place_in`	containment placement	200k	800

TABLE V: **Skill-to-checkpoint routing and per-skill budgets used in all experiments.** All checkpoints are $\pi_{0.5}$ models finetuned from a shared 50k-step all-skills initialization; skills in the same group share a checkpoint.

Additional Diagnostics

This appendix collects the per-rollout diagnostics behind the main-text attribution.

Diagnostic	Reached-area	Arm-reach
Tasks solved (q=1)	0 / 10	1 / 10
Mean task score $\bar{q}$	0.01	0.10
`move_to` attempts	74	99
Next-skill readiness failures	23	35
Target grounding failures	31	37
Control / commit failures	61	58
Total failed attempts	115	130

TABLE VI: **Effect of the arm-reach handoff criterion in the navigation verifier** (live VLA, K=200, a representative round, one official instance per task). Both columns use the same tasks, instances, and checkpoints; only the navigation verifier criterion differs (reached-area vs. arm-reach), making this a controlled ablation. Tightening the navigation postcondition from *reached the target area* to *the target is within arm reach* reclassifies handoff stalls: it surfaces 12 additional next-skill readiness failures and triggers 25 more `move_to` attempts (re-navigation to a reachable pose), while control/commit execution stays the binding cause. It also recovers the radio task, whose toggle only succeeds once the robot is forced to re-navigate to a reachable pose. Counts are trace-derived diagnostics from structured verifier reasons (cf. Table III), not ground-truth labels.

Table VI reports the effect of tightening the navigation handoff criterion to an arm-reach test, which reclassifies handoff stalls and recovers the radio task.

The completed ablation is the arm-reach handoff criterion (Table VI). Three further ablations run on the same task suite, instances, and checkpoints but are left to future work: a no-recovery versus replanning comparison that separates VLA skill brittleness from recovery-policy limits, a verifier-cadence sweep over $K \in \{50, 100, 200\}$, and a next-skill-conditioned readiness predicate that verifies $\phi_t \wedge \rho$ directly rather than through `move_to` wording (Sec. V-C).

Table VIII gives the per-task attempt, success, verifier-call, and termination counts behind these diagnostics.

Verifier example. The verifier is queried with the skill's natural-language postcondition, plus an arm-reach clause for `move_to`. On the radio rollout, the first `move_to(radio)` check, logged under the raw token `navigate_to`, asks: *"Has the robot successfully completed this skill: 'navigate to the radio'? Judge only from the current camera views. For navigate_to, success requires the navigation target to appear close-up and prominent in the head camera at arm-reach distance; if it is only visible far away, small, or across the room, answer no."* It returns `verdict=no`, `confidence=0.8`, with rationale *"the head camera does not show any object identifiable as a radio close-up and prominent at arm-reach distance… the robot has not successfully navigated to the radio,"* and `recovery_hint` *"re-localize or search for the radio."* This readiness verdict triggers re-navigation, and the task later succeeds once the radio is reached at arm-reach—the re-plan/retry/advance loop visualized in Fig. 1.

Metric	Value
Audited verifier failures (2-annotator consensus)	21
Confirmed real failures (human)	20
Failure-decision reliability	0.95
Over-strict rate (real success wrongly failed)	0.05
Inter-annotator agreement	90%

TABLE VII: **Blinded human audit of the verifier's failure decisions.** Two authors independently judged sampled verifier decisions from the same head/wrist views, blind to the verdict; they agreed on 90% of non-*unclear* items. Of the verifier's *failure* decisions they could adjudicate (21 in consensus)—the decisions that underlie the failure attribution in Table III—20 were confirmed as real failures, an over-strict rate of 0.05. A reported failure is thus very likely a real failure rather than a verifier artifact. The audit validates the binary failure decision, not the failure *category*, which remains verifier-derived; N is small and these are preliminary indicators.

Verifier audit. Because the failure attribution is read off the verifier, we audit its failure decisions. Two authors independently judged sampled verifier decisions from the same head and wrist views, blind to the verifier verdict, agreeing on 90% of non-*unclear* items. Of the verifier's failure decisions they could adjudicate in consensus (21), 20 were confirmed as real failures—an over-strict rate of 0.05, so the verifier almost never rejects a skill that actually succeeded and a reported failure is a real failure. Task-level success is independent of this audit, being scored by the BDDL predicate. The audit covers the binary failure decision only, not the failure category, and N is small (Table VII).

Proposed Readiness Template Library

Table IX lists the proposed next-skill readiness predicate templates $R_{\text{name}(s_{t+1})}$ for instantiating $\rho(s_t, s_{t+1})$ (Sec. V-C). Each is indexed by the next skill type and is a skill-contract rule, not a per-task pairwise hand-written condition; the current system approximates them through `move_to` postcondition wording rather than verifying $\phi_t \wedge \rho$ directly.

Task	Attempts	Succ.	Fail	Frames	VLM Calls	End	q
Turn on radio	5	3	2	2 160	4	success	1
Pick up trash	20	9	11	10 535	16	timeout	0
Make microwave popcorn	14	9	5	6 475	12	timeout	0
Move boxes to storage	44	13	31	27 900	39	finish	0
Set mousetraps	27	9	18	16 000	23	finish	0
Hide Easter eggs	23	9	14	10 800	21	finish	0
Freeze pies	33	15	18	23 180	32	finish	0
Cook hot dogs	28	15	13	18 289	22	timeout	0
Bring in wood	16	15	1	7 100	10	finish	0
Put shoes on rack	22	3	19	15 384	17	timeout	0
Total (10 tasks)	232	100	132	137 823	196	—	1/10

TABLE VIII: **Per-task end-to-end trace diagnostics** (live VLA, $K = 200$, a representative round, one official instance per task). All ten BEHAVIOR-1K tasks supported by the current skill library are covered with one rollout each. Succ. and Fail count skill attempts (not task outcomes); Frames is the total environment steps consumed across all attempts; VLM calls is the cumulative verifier query count; End is the rollout termination reason (`success`/`timeout`/`finish`) and q the BEHAVIOR-1K task-predicate score. Radio is the one solved task under the arm-reach handoff criterion; of the 132 raw failed attempts, two are verifier-side cases excluded from the attribution in Table III (130).

Next Skill	Readiness Predicate Template
`pick_up_from(x,y)`	$\text{visible}(x) \wedge \text{graspReachable}(x)$; merely visible at a distance is insufficient.
`open_door(a)`	$\text{visible}(\text{handleOrEdge}(a)) \wedge \text{manipReachable}(a)$.
`close_door(a)`	$\text{doorOpen}(a) \wedge \text{visible}(\text{handleOrEdge}(a)) \wedge \text{manipReachable}(a)$.
`press(a)`	$\text{visible}(\text{controlRegion}(a)) \wedge \text{manipReachable}(a)$.
`turn_on_switch(a)`	$\text{visible}(\text{controlRegion}(a)) \wedge \text{manipReachable}(a)$.
`place_in(x,c)`	$\text{held}(x) \wedge \text{containerReachable}(c) \wedge \text{openingPlausible}(c)$.
`place_on(x,s)`	$\text{held}(x) \wedge \text{supportReachable}(s)$.
`place_on_next_to(x,s)`	$\text{held}(x) \wedge \text{supportReachable}(s)$.
`move_to(x)`	no additional predicate beyond ϕ_t; subsequent manipulation steps impose their own readiness predicate.

TABLE IX: ***Proposed*** **next-skill readiness templates for instantiating** $\rho(s_t, s_{t+1})$ **(Sec. V-C).** They are skill-contract templates, not per-task pairwise hand-written rules. In the current system readiness is approximated through `move_to` postcondition wording rather than these templates; verifying $\phi_t \wedge \rho$ with a next-skill-conditioned verifier is future work.