

MIND: Instruction-Conditioned Incremental Grounding for Robots

Mohamed Abouagour¹, Junhyun Nam², Wonse Jo², and Byung-Cheol Min¹

Abstract—Robots following persistent language instructions must decide which cached room judgments remain valid as their environment changes. We present MIND (Maintenance via INstruction-conditioned Dependencies), a framework for maintaining target-room decisions without re-evaluating every room after each event. A language model compiles each instruction into dependencies for semantic qualification and target selection. The controller intersects semantic dependencies with event footprints, refreshes affected judgments, and updates reachability and selection separately. For fixed deterministic evaluation, planning, and selection, conservative dependency and footprint coverage preserves the decisions of full recomputation. On 240 five-event ProcTHOR sequences with an exact benchmark predicate, MIND reduces mean semantic evaluations from 30.6 to 11.2 while preserving all 1,200 post-event decisions. Frozen-evaluator experiments with Phi-4 and Qwen3-14B also preserve all 1,200 decisions, without improving the evaluators’ underlying task accuracy. In 32 closed-loop AI2-THOR episodes using exact-predicate qualification and metadata-derived footprints, MIND matches the final target decision of full recomputation in every episode. Unitree G1 experiments demonstrate target reuse and reselection across room transitions without additional semantic evaluations under MIND, and abstention after a barrier and an object removal at 4 evaluations against 15 to 18. Additional materials are available at <https://mind-icra27.github.io/>.

I. INTRODUCTION

A robot may continue following the same instruction while its environment changes. For “bring me the nearest mug,” entering another room, closing a doorway, or removing an object can alter the appropriate target in different ways. The robot has already evaluated which rooms satisfy the instruction, yet those cached judgments need not all become invalid together. Re-evaluating every room discards reusable computation; retaining every judgment can preserve a stale target. Each room judgment is a language-model call, which on a robot shares the GPU with perception and on a hosted model is billed, so the calls avoided are the resource at stake. We ask: *after a world change, which room judgments must the robot check again?*

The event alone does not determine the required update. For *the nearest chair*, robot motion can change the selected room without changing which rooms contain chairs. For *a chair in the room I am in*, the same motion changes which rooms qualify. The same variable thus enters qualification or selection depending on the instruction, and its stage decides whether semantic re-evaluation is needed.

Fig. 1 illustrates the distinction within one persistent instruction. Moving to another room changes the ranking of

Instruction: “bring me the nearest mug”

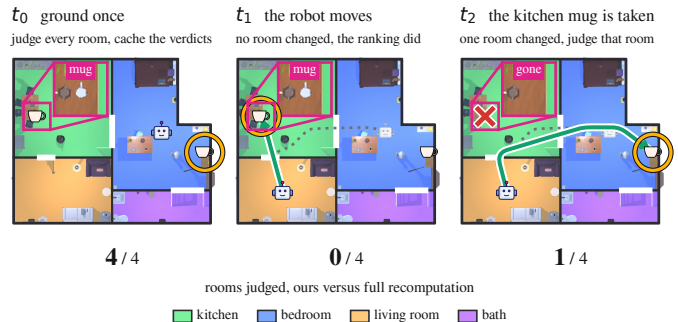


Fig. 1: An overview of “Bring me the nearest mug.” At t_0 every room is judged and cached. Moving (t_1) changes no room: judgments are reused, only the target is re-chosen. Removing the kitchen mug (t_2) refreshes one judgment. Full recomputation rejudges all four each time.

candidates but not their inventory-based qualification, so no room judgment needs refreshing. Removing the kitchen mug instead requires rechecking the kitchen while retaining the other judgments.

Dependency-based reuse is well established in incremental computation, where dependencies may be declared in program or query structure or discovered during execution [1–4]. In our setting, the state dependencies relevant to maintenance are implicit in a language instruction and differ between qualification and selection. MIND infers these instruction-specific dependencies from language, separately for qualification and selection, to determine which cached judgments require updating. We therefore study decision-preserving maintenance of a given grounding pipeline, rather than replacing its semantic evaluator or replanning the entire task after every event.

We present MIND (Maintenance via INstruction-conditioned Dependencies), a framework for *instruction-conditioned incremental grounding*. Before execution, a language model compiles the instruction into separate dependency sets for room qualification and target selection. After an event, the controller intersects the semantic dependencies with a footprint of reported state changes to identify which cached judgments to refresh. Reachability and deterministic selection are maintained separately, allowing a target to change without unnecessary semantic evaluation. Under the fixed-evaluator and conservative-coverage assumptions of Section IV, MIND preserves the decisions of full recomputation. Over-approximation may increase work, whereas missing coverage can leave stale judgments; the guarantee does not certify the compiler’s predictions or the evaluator’s accuracy. We contribute:

¹Indiana University Bloomington, USA. ²Incheon National University, Republic of Korea.

- Instruction-conditioned, stage-specific dependency signatures combined with region-scoped event footprints, separating semantic refresh, reselection, and reuse.
- Sufficient conditions under which this maintenance preserves full-recomputation decisions, with semantic cost as initialization plus later regional refreshes.
- On 240 five-event ProcTHOR sequences, mean semantic evaluations fall from 30.6 to 11.2 with all 1,200 decisions unchanged; compiler, footprint, and interface tests examine failures outside the guarantee, 32 closed-loop AI2-THOR episodes test maintenance during execution, and 26 Unitree G1 runs over four scenarios exercise reuse, reselection, global refresh, and abstention on hardware.

II. RELATED WORK

Grounded planning and dynamic revision. Language-conditioned robots ground instructions in affordances and scene graphs [5–7], revise plans from feedback [8–10], and abstain or relax infeasible goals [11–13]. Other systems maintain contextual or task memory for replanning [14–17]; Event-Grounding Graph represents spatial structure and events [18], while LINGO-Space updates location distributions as referring expressions arrive [19]. These approaches motivate maintaining useful world knowledge, but stored information can become stale [20]. MIND addresses the validity of cached room-qualification judgments under a persistent instruction: which judgments a world change requires the controller to refresh, and when selection alone must be reconsidered.

Incremental computation and embodied reuse. Self-adjusting computation, view maintenance, build systems, and truth maintenance reuse outputs through dependencies on changed inputs [1–4]. Dependencies may be declared in formal descriptions or tracked during execution. KEEP manages model-state reuse, whereas AgenticCache caches and validates planning transitions [21, 22]. MIND shares the dependency-based reuse principle but caches regional semantic verdicts rather than model states or plan transitions: instruction-conditioned signatures separate qualification inputs from selection inputs, regional footprints scope the invalidation, and the rule preserves a fixed pipeline’s decisions when both conservatively cover the required updates.

Dependencies derived from language. LiP-LLM, DeComplex, and DART-LLM infer precedence among subtasks [23–25]; incremental parsing repairs grammar or lexical knowledge after parsing or planning failures [26]. LG-TAP constructs temporally compact world models through backward observation search and uses efficient graph updates to reuse grounding computations by selectively recomputing world-dependent factor features [27]. MIND applies dependency-based reuse to a different update unit: cached room-qualification verdicts supporting a persistent target decision. It infers separate qualification and selection dependencies and combines them with event footprints, yielding explicit conditions for decision-preserving regional updates under conservative estimates.

III. PROBLEM FORMULATION

System and event stream. An episode fixes a natural-language instruction q and a building with regions $\mathcal{R}$; in our indoor experiments each region is one room. The episode then evolves through events $e_{1:T}$. Event e_t changes the world from X_{t-1} to X_t , where X_t holds region inventories, semantic region labels, the robot’s region, interaction history, and the region-adjacency graph. We denote these state categories by

$$\mathcal{V} = \{\text{inventory, region label, position, history, topology}\}. \quad (1)$$

We instantiate $\mathcal{V}$ with five state categories common to indoor grounding. Concretely, inventory is the multiset of object types present in a region, so counts matter and removing one of several chairs is a change, region label is its semantic name, position is the region the robot currently occupies rather than a metric pose, history is the set of regions already used to satisfy the instruction, and topology is the region-adjacency graph with per-doorway passability. History is therefore region-level: a request such as *another chair* asks for a chair in a region not yet used, not for a different chair instance within a used region, which the state does not represent. Grounding remains region-level and does not represent object coordinates or instance identities. In the graph benchmarks, *nearest* means fewest doorway hops in G_t ; the closed-loop AI2-THOR experiment instead uses navigation-grid path length. The rule requires only a variable vocabulary shared by compiler, evaluator, and detector; further categories can be added without changing it. We assume their current values are available to the controller; perception is outside the present formulation. The system contains a semantic evaluator, a reachability planner, a deterministic selector with fixed tie-breaking, and a controller that maintains their results across events.

A fresh semantic evaluation, the room-level semantic check, asks whether region r satisfies q in the current state,

$$s_q(r; X_t) \in \{0, 1\}. \quad (2)$$

Let G_t denote the region-adjacency graph at time t , let $c_t \in \mathcal{R}$ be the robot’s current region, and let T be the number of events. The planner returns the reachable-region set $R_t = \text{Reach}(G_t, c_t)$. For any binary regional support vector z , define

$$\text{Feas}(z, R_t) := \{r \in \mathcal{R} : z(r) = 1\} \cap R_t. \quad (3)$$

Full recomputation evaluates every region and returns the feasible set and the selected target region, or NONE,

$$\begin{aligned} F_t^{\text{full}}(q) &= \text{Feas}(s_q(\cdot; X_t), R_t), \\ a_t^{\text{full}} &= \pi_q(F_t^{\text{full}}(q); X_t), \end{aligned} \quad (4)$$

where the request-specific selector π_q may rank feasible candidates using current state and returns NONE when none remain.

Cache of room judgments and objective. The incremental controller stores $\hat{S}_{q,t}(r)$, its most recent semantic verdict for region r , initialized exactly as $\hat{S}_{q,0}(r) = s_q(r; X_0)$ for every

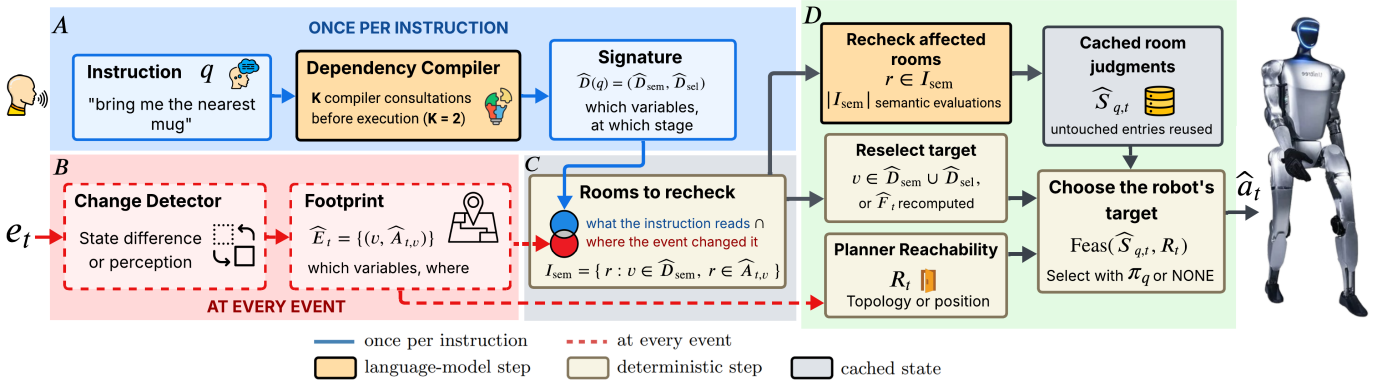


Fig. 2: **Architecture for “bring me the nearest mug.”** (A) Once per instruction, the compiler separates variables needed to judge each room from those needed only to choose among qualifying rooms. (B) After each event, the detector reports what changed and which rooms may be affected. (C) Their intersection fixes which room judgments to recheck; position and topology changes also update reachability. (D) The controller reuses untouched judgments and reselects a reachable target whenever the feasible set is recomputed or a reported change intersects either compiled dependency slot.

region. After event e_t , a maintenance policy chooses a refresh set $U_t(q) \subseteq \mathcal{R}$ and updates

$$\hat{S}_{q,t}(r) = \begin{cases} s_q(r; X_t), & r \in U_t(q), \\ \hat{S}_{q,t-1}(r), & r \notin U_t(q). \end{cases} \quad (5)$$

It maintains $\hat{F}_t(q) = \text{Feas}(\hat{S}_{q,t}, R_t)$ and reruns π_q whenever the feasible set or a selection input may have changed; $\hat{a}_t$ is the resulting decision. For $n = |\mathcal{R}|$, the semantic-evaluation cost is

$$C_{\text{sem}} = n + \sum_{t=1}^T |U_t(q)|, \quad (6)$$

against $n(T+1)$ for full recomputation. The problem is

$$\min C_{\text{sem}} \quad \text{subject to} \quad \hat{a}_t = a_t^{\text{full}} \text{ for every } t. \quad (7)$$

Planner and deterministic-selection work is accounted for separately.

Scope. Semantic qualification is unary in the candidate region, and its inventory dependence is restricted to that region’s contents. Relations to a distinguished state-dependent region, such as the robot’s current region, remain unary in the candidate, but predicates that inspect other regions’ inventories are outside the locality assumption used below. Predicates over multiple free candidate regions, cross-candidate cardinality constraints, and future-state prediction are outside scope.

IV. METHODOLOGY

MIND compiles dependencies once per instruction and uses event footprints to maintain cached room judgments, reachability, and target selection (see Fig. 2).

A. Instruction dependency map

The compiler (Fig. 2) maps q to

$$\hat{D}(q) = (\hat{D}_{\text{sem}}(q), \hat{D}_{\text{sel}}(q)), \quad (8)$$

The true read sets are defined counterfactually for the fixed semantic function s_q and selector π_q used in the comparison. A variable v belongs to $D_{\text{sem}}^*(q)$ if changing only state category v can change $s_q(r; X)$ for some candidate region. It belongs

to $D_{\text{sel}}^*(q)$ if, with the feasible set fixed, changing only v can change the selected region. The two sets need not be disjoint: a variable may affect both qualification and selection. A selection-only change requires deterministic reselection but no fresh semantic verdict. These true read sets are unavailable to the controller. Reachability is maintained separately, is refreshed whenever topology or robot position changes, and is always applied through (3).

The compiler prompt defines the five categories, explains the two slots, and supplies one development example per request family. It emits two variable lists, parsed into (8), with inventory included in the semantic slot by construction. Decoding is greedy. The frozen downstream configuration compiles under K cyclic orders of the six examples and unions each slot, with $K=2$ unless stated; these are K pre-execution consultations per instruction, never repeated after world events.

Two repairs expand a signature monotonically. *Sample union* unions each slot over the K orders. *Stage widening* treats reported variables as possibly belonging to either stage. With $W = \hat{D}_{\text{sem}} \cup \hat{D}_{\text{sel}}$, define

$$\text{widen}(\hat{D}_{\text{sem}}, \hat{D}_{\text{sel}}) = (W, W \setminus \{\text{inventory}\}). \quad (9)$$

The selector never reads inventory. Widening repairs semantic-to-selection demotions and preserves (12) whenever the unwidened signature satisfies it, at the cost of additional semantic evaluations. When widening is evaluated, it is applied after sample union. Unless explicitly labeled as widened, the main configuration uses slot-wise union only ($K=2$); widened configurations are reported separately. Neither operation recovers a variable absent from every compiler output.

B. Event footprint

The detector represents an observed event by

$$\hat{E}_t = \{(v, \hat{A}_{t,v})\}, \quad (10)$$

where $v \in \mathcal{V}$ is a reported changed variable and $\hat{A}_{t,v} \subseteq \mathcal{R}$ is its estimated invalidation scope. Reference footprints use changed rooms for inventory: one for a removal, source and destination for a relocation. Position, topology, history, and region label

use $\mathcal{R}$ whenever they change, conservatively allowing nonlocal effects on qualification. This is an invalidation convention, not a claim that every room changes physically. Detector estimates may miss or add variables or regions; Proposition 1 needs coverage of the reference footprint.

C. Scoped update and robot target selection

Intersecting (8) and (10) yields

$$I_{\text{sem}}(e_t, q) = \{r \in \mathcal{R} : \exists(v, \hat{A}_{t,v}) \in \hat{E}_t, v \in \hat{D}_{\text{sem}}(q), r \in \hat{A}_{t,v}\}. \quad (11)$$

The policy sets $U_t(q) = I_{\text{sem}}(e_t, q)$ and then:

- 1) refreshes planner outputs if topology or robot position was reported changed;
- 2) updates (5) only for $r \in U_t(q)$;
- 3) recomputes $\hat{F}_t(q)$ if reachability changed or $U_t(q) \neq \emptyset$; and
- 4) reruns π_q if $\hat{F}_t(q)$ was recomputed or a reported changed variable lies in $\hat{D}_{\text{sem}}(q) \cup \hat{D}_{\text{sel}}(q)$.

Only step 2 invokes the semantic evaluator. The downstream steps trigger on a report rather than only on a changed verdict: a selection dependency that the compiler promoted into the semantic slot may refresh no region and reproduce every support value, and step 4 still reruns selection because the variable was reported changed.

D. Conservative estimates and correctness

The controller receives $\hat{D}(q)$ and $\hat{E}_t$, and either may be inexact. We say $\hat{D}(q)$ *conservatively contains* $D^*(q)$ when

$$D_{\text{sem}}^* \subseteq \hat{D}_{\text{sem}} \quad \text{and} \quad D_{\text{sel}}^* \subseteq \hat{D}_{\text{sem}} \cup \hat{D}_{\text{sel}}. \quad (12)$$

Let E_t^* contain all actually changed categories, using the preceding footprint convention: changed rooms for inventory and $\mathcal{R}$ for the other categories. Define $\hat{E}_t \succeq E_t^*$ when every $(v, A_{t,v}^*) \in E_t^*$ has a corresponding $(v, \hat{A}_{t,v}) \in \hat{E}_t$ with $A_{t,v}^* \subseteq \hat{A}_{t,v}$.

With the semantic evaluator and its input interface fixed, approximation errors are asymmetric. Promoting a selection dependency into the semantic stage causes unnecessary re-evaluation but still triggers reselection; demoting or omitting one can leave stale support and voids the guarantee, though it need not change the target on every event. Adding dependencies or regions preserves deterministic outputs.

Proposition 1 (Decision equivalence under dual conservative containment): Assume $\hat{S}_{q,0} = s_q(\cdot; X_0)$; all task-relevant changes are in $\mathcal{V}$; inventory dependencies are local to the candidate region; the semantic function s_q is a fixed function of the state, chosen independently of $\hat{D}$ (the reference-field interface of Section V); (12) holds; every observed footprint satisfies $\hat{E}_t \succeq E_t^*$ under the stated footprint convention; and semantic evaluation, planning, and selection are deterministic. Then $\hat{a}_t = a_t^{\text{full}}$ after every event.

Proof sketch. Induct on events. The caches agree initially. If a fresh verdict $s_q(r; X_t)$ can differ from $\hat{S}_{q,t-1}(r)$, some variable in D_{sem}^* changed with r in its true footprint. Signature

and footprint containment place r in (11), so it is refreshed; thus no stale semantic value survives and $\hat{S}_{q,t} = s_q(\cdot; X_t)$. Observed topology and position changes also refresh planner output, so $\hat{F}_t = F_t^{\text{full}}$. Finally, any changed variable in D_{sel}^* lies in $\hat{D}_{\text{sem}} \cup \hat{D}_{\text{sel}}$, so step 4 reruns selection whenever it is reported changed, whether or not its footprint refreshed a region. Extra reported dependencies or regions add work but cannot alter a deterministic result. $\square$

Proposition 1 preserves a fixed evaluator’s decisions, not their task correctness. Compiled prompt fields change the evaluator itself, so agreement within that interface need not imply agreement with the reference-field evaluator; Section V defines both interfaces and Table III reports task accuracy separately.

Semantic-evaluation cost. For $m_t = |U_t(q)|$, the cost is $n + \sum_t m_t$ in (6); savings vanish when global refreshes give $m_t \approx n$. Maintaining all cached verdicts is stronger than preserving only the target, so this is not a minimum-cost solution to (7).

V. BENCHMARKS AND EXPERIMENTAL SETUP

The six-family dependency and request benchmarks use automatically checked manifests frozen by hash before model evaluation. A dependency manifest is refused if a sequence has no verdict-changing event or abstains throughout, a history sequence is not moved by its use event, a move or door event changes a verdict for a nearest request, or no event exposes the inventory-only rule; a request manifest is refused if families are unbalanced or share a signature, a minimal pair adds an edge other than the one claimed, a held-out template leaks into development, or a signature disagrees with the event instrument. These are consistency checks, not independent certification of natural-language semantics. Generators, prompts, and model configurations are in the repository¹.

Dependency benchmark. The primary benchmark contains 240 five-event sequences over houses from the ProcTHOR [28] test split, using 11 distinct instructions across six request families. *Inventory-only* reads inventory only; *named-region* adds a semantic region label; *current-region* adds robot position to qualification; *history-dependent* adds the set of previously used regions; *nearest-candidate* places position and topology in selection; and *nearest+history* combines the last two. Events shut doorways, empty or relocate objects, move the robot, and consume a referent. After each event, the manifest records the reference target and the regional verdicts that actually change value. The latter define invalidation recall and precision, distinguishing missed changed verdicts from conservative extra refreshes. The 240 sequences span 57 houses, with 55, 9, 27, 54, 40, and 55 sequences for the six families in the order listed. The 11 instructions are held-out-template phrasings from the request benchmark and are disjoint from the six compiler examples; two also appear among the minimal pairs.

Request benchmark. The request benchmark has 216 requests from 36 templates, six per family, and holds out whole templates rather than sentences, since a sentence split measures

¹<https://anonymous.4open.science/r/mind-icra27>

rewording; the frozen compiler evaluation uses the 69 requests of the held-out templates. It also includes 144 minimal pairs in which a modifier changes a specified set of stage-specific dependency edges while leaving unrelated edges unchanged.

Unseen-placement benchmark (Adjacent). A seventh family tests a dependency placement absent from all six compiler demonstrations. The benchmark interprets *a chair in a room next to the one I am in* as a chair in a room connected to the robot’s current room by one currently passable doorway, so closing that doorway changes qualification although the rooms remain physically adjacent. The semantic dependencies are inventory, position, and topology. The judge receives passability as a state field; the compiler sees only the phrasing and the variable definitions, in which topology is reachability given doors. The benchmark contains 80 sequences in 80 houses, six held-out phrasings, three of which name a door or connection explicitly, and all five event types.

Backbones. We evaluate Phi-4, Qwen3-8B and Qwen3-14B (microsoft/phi-4, Qwen/Qwen3-8B, Qwen/Qwen3-14B) as locally deployable models through HuggingFace Transformers, and gpt-oss-120b through a hosted OpenAI-compatible endpoint. Decoding is greedy (temperature 0 at the endpoint), Qwen3 thinking mode is disabled, compiler outputs are capped at 512 new tokens, and judge verdicts at 8. Local weights load in bfloat16 on one L40S GPU (Transformers 5.8, PyTorch 2.10), and judge prompts run in chunks of four per forward pass. The fallible-evaluator experiments use frozen Phi-4 and Qwen3-14B verdicts; the timing study uses Phi-4.

Policies. Within each evaluator-interface condition, policies share the semantic evaluator, planner, deterministic selector, and composition rule (3). They differ in the maintenance schedule, including semantic refreshes and reselection triggers. Table I compares full recomputation, a static inventory-only signature, a universal semantic signature, reference variables with and without stage placement, and the compiled signature. Reference signatures are oracle controls, not inferred baselines. The exact predicate, the affordance sets it scores, and the selector are benchmark components fixed per family: the selector returns the fewest-hop feasible region for nearest-candidate requests, breaking ties on the display name, and the alphabetically first feasible region otherwise. Every policy reruns this selector after every event, so the inventory-only row fails only through stale semantic verdicts and never through suppressed reselection. Maintenance policies receive the compiled signature and the footprint, not the family label.

Semantic evaluators and prompt interfaces. An exact benchmark predicate isolates maintenance errors from semantic-model errors. Fallible-judge experiments use a *frozen verdict tape*: each distinct regional question is evaluated once per interface, and the resulting verdict is replayed to every policy; an unparsable generation aborts the run. The *reference-field interface* constructs prompts from $D_{\text{sem}}^*(q)$ and holds these fields fixed independently of the predicted signature. A prompt carries exactly one line per declared variable: the candidate region’s contents for inventory, and position, history, and adjacency as yes/no relations to that region; the tape key

is the full prompt string. The *compiled-field interface* instead uses $\widehat{D}_{\text{sem}}(q)$ without reference fields. An omitted field can make both policies agree on an incorrect predicate, so task accuracy is evaluated separately from agreement. Phi-4 recovers the semantic slots exactly on the 11 main dynamic instructions, making the two interfaces coincide there; the unsafe-signature and Adjacent experiments test their divergence.

Metrics. For five-event sequences, *episode accuracy* requires all five post-event targets to match the benchmark reference. *Incremental agreement* measures target agreement with full recomputation under the same evaluator and interface. *Semantic evaluations* count regional predicate evaluations including initialization, as in (6); under a frozen tape these are logical refresh costs, not repeated inference calls. *Conservative containment* is (12), and *semantic inflation* is $|\widehat{D}_{\text{sem}}|/|D_{\text{sem}}^*|$.

VI. RESULTS

Every number below is measured on the frozen evaluation units of Section V with the metrics defined there. Semantic-evaluation counts are reported per five-event sequence. Table I reports episode-accuracy proportions with 95% Wilson intervals and semantic-evaluation means with standard deviations. Decision-agreement rates are descriptive proportions over the evaluated decisions; the observation-derived footprint experiment reports a one-sided Clopper–Pearson bound.

A. Ablation: the same targets, fewer rooms rechecked

Conservative coverage and update scope. Table I separates target accuracy from semantic cost using the exact predicate. The static inventory-only policy attains 0.479 episode accuracy and 0.703 invalidation recall, whereas all containing signatures preserve the 1,200 decisions. Along the ablation sequence, regional footprints reduce mean evaluations from 30.6 to 23.4, request-specific variables to 14.1, and stage separation to 11.2; these are sequential differences, not order-independent contributions. Stage separation matters for the families with selection dependencies: $18.8 \rightarrow 8.3$ (*nearest-candidate*) and $23.5 \rightarrow 13.2$ (*nearest+history*).

TABLE I: Maintenance ablation on 240 five-event robot episodes with the exact benchmark predicate. Episode accuracy with 95% Wilson intervals; semantic evaluations per sequence, mean $\pm$ SD.

policy	episode acc.	sem. evals	inv. recall
full recomputation	1.000 [.984, 1]	30.6 $\pm$ 9.7	1.000
static inventory-only	0.479 [.417, .542]	8.1 $\pm$ 1.6	0.703
universal semantic signature	1.000 [.984, 1]	23.4 $\pm$ 6.4	1.000
reference variables, no stage split	1.000 [.984, 1]	14.1 $\pm$ 6.6	1.000
reference stage-specific	1.000 [.984, 1]	11.2 $\pm$ 3.6	1.000
MIND, Phi-4 ($K=2$)	1.000 [.984, 1]	11.2 $\pm$ 3.6	1.000

Compile-once signatures outperform event-wise relevance decisions. The compiler-free alternative shows the instruction and each event to a language model and asks whether recomputation is needed (NONE, SELECTION, or SEMANTIC), with the compiler’s six examples and stage definitions in the prompt; everything else is shared. Compile-once signatures reach 1.000 episode accuracy at 11.2 evaluations from 22 compiler consultations with no non-containing decision.

Event-wise relevance reaches 0.658 at 13.1 evaluations: of 1200 decisions, 184 omit a required semantic dependency and 165 add a spurious one. The Phi-4 decider is asked once per event (1200 logical calls, 434 distinct prompts since identical prompts are replayed) and returns different verdicts for 18 of the 55 instruction and event-type pairs recurring with other room names.

B. Instruction-conditioned compilation for new requests

Containment versus exact recovery. On the 69 held-out requests (Table II), Phi-4 and Qwen3-14B produce only conservative errors across six demonstration orders. Qwen3-14B attains containment 1.000 ± 0.000 despite exact recovery 0.454 ± 0.097 ; Qwen3-8B attains 0.988 ± 0.019 containment. gpt-oss-120b has the highest exact recovery (0.935), without inflation or non-containing outputs.

Sample union. For Qwen3-8B, increasing K from 1 to 3 raises containment from 0.988 to 1.000 and removes all non-containing outputs, but lowers exact recovery from 0.372 to 0.210 at 21% inflation. Means are over six demonstration orders.

TABLE II: Dependency-compiler evaluation on the frozen 69-instruction held-out set.

compiler, K	exact sig.	containment	inflation	non-containing (total)
gpt-oss-120b, $K=1$	0.935	1.000	1.00	0
Phi-4, $K=1$	0.727	1.000	1.04	0
Phi-4, $K=2$ (deployed)	0.667	1.000	1.06	0
Qwen3-14B, $K=1$	0.454	1.000	1.08	0
Qwen3-14B, $K=2$	0.353	1.000	1.11	0
Qwen3-8B, $K=1$	0.372	0.988	1.12	5
Qwen3-8B, $K=2$	0.273	0.998	1.17	1
Qwen3-8B, $K=3$	0.210	1.000	1.21	0

On the 144 minimal pairs, scored by whether the variant’s signature gains exactly the modifier’s variables in the right slot (edge-correct) and nothing in the other slot (clean), Phi-4 reaches 0.854 and 0.799, gpt-oss-120b 0.743 and 0.729, Qwen3-14B 0.681 and 0.465, and Qwen3-8B 0.639 and 0.569. The hard case is the modifier that should add position and topology to *selection*: edge-correct for 0 of 36 pairs under both Qwen models, which place it in the semantic slot (0.58 for Phi-4, 0.47 for gpt-oss-120b), a conservative promotion matching the no-stage-split ablation.

Conservative repair on Adjacent. For Adjacent’s passable one-hop rule (Section V), only gpt-oss-120b places topology in the semantic slot across all six demonstration orders (raw containment 1.000); Qwen3-14B, Phi-4, and Qwen3-8B place it in selection (0.722, 0.556, 0.444). Stage widening (9) repairs the demotion to 1.000 containment for the first two, and Qwen3-8B additionally needs a $K=3$ union to recover omitted variables. With the exact benchmark predicate, every repaired signature reaches 1.000 episode accuracy at 18.3 semantic evaluations, compared with 1.000 at 30.4 for full recomputation. With a Phi-4 judge over 400 decisions, the raw Phi-4 signature yields agreement of 0.960 under the reference-field interface and 0.953 under the compiled-field interface; widening restores agreement on all 400 decisions under both interfaces. The $1.66\times$ reduction is smaller than in the primary

benchmark because position and topology affect qualification and trigger global refreshes. Full-recomputation episode accuracy remains 0.06 under the compiled-field interface, whose prompt fields come from the raw, unwidened Phi-4 signature, and 0.00 under the reference-field interface, with final-step accuracy of 0.38 and 0.31, respectively. Widening, adopted after these demotions were observed, reads no labels; on the primary benchmark it keeps every decision at 15.3 evaluations.

TABLE III: Interface ablation: a missed dependency and its conservative repair (Phi-4 judge, 1200 decisions).

compiler signature	interface	agreement	episode acc.	evals
Qwen3-8B, one sample	reference	0.992 (1190/1200)	0.329	10.4
Qwen3-8B, one sample	compiled	1.000	0.329	10.4
Qwen3-8B, $K=2$ union	reference	1.000	0.342	11.5
Phi-4, $K=2$ (main)	both	1.000	0.342	11.2

C. Effects of missed dependencies and footprints

Missed dependencies and repair. A Qwen3-8B signature omits semantic history (Table III). Under the reference-field interface, stale reuse changes 10/1200 decisions, reducing agreement to 0.992 and episode accuracy from 0.342 to 0.329, while evaluations fall from 11.2 to 10.4. Under the compiled-field interface, both policies lack history and agree on all 1200 decisions, but both score 0.329 episode accuracy. The same three history episodes fail through different mechanisms: stale reuse versus an incomplete evaluator input. A $K=2$ union restores reference-field agreement on all decisions and accuracy 0.342 at 11.5 evaluations.

Approximate event footprints. Perturbing the footprint alone over 240 sequences with the exact predicate, over-approximation never costs correctness (agreement, accuracy, and recall stay 1.000 while evaluations rise from 11.2 to 18.5 when every untouched region is added); under-approximation degrades monotonically as evaluations fall to 8.2, its stale-step rate tracking the fraction of partial footprints (Pearson 0.997). Mislabeling the changed variable behaves the same way: acting on the detector’s top guess falls to 0.683 accuracy at a 40% error rate, whereas invalidating over the whole candidate set holds 1.000 at 27.5 evaluations versus 30.6.

D. Closed-loop and observation-derived robot updates

Inferred maintenance matches full recomputation under fallible evaluators. With frozen Phi-4 and Qwen3-14B evaluators, inferred maintenance matches full recomputation on all 1200 post-event decisions across 240 sequences at 11.2 semantic evaluations versus 30.6. The static inventory-only rule reaches agreement 0.958 and 0.983, respectively. Full-recomputation episode accuracy is 0.342 with Phi-4 and 0.275 with Qwen3-14B. That statistic compounds evaluator error over five decisions (final-step accuracy is 0.704 and 0.638); inferred maintenance reproduces both figures exactly, so agreement measures maintenance fidelity, not semantic-model competence.

Selective maintenance reduces semantic-evaluator runtime. On the separate 59-sequence timing subset, full recomputation averages 29.2 semantic evaluations and 1.44s of Phi-4

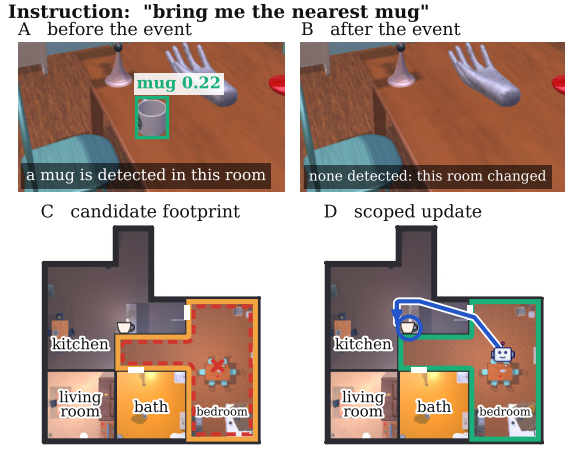


Fig. 3: **Removal localization** in ProcTHOR validation house 571. (A, B) Matched views before and after mug removal. (C) The candidate footprint contains the affected bedroom, the only room refreshed here. (D) Reselection identifies the kitchen mug; the living room remains unreachable.

evaluator time per five-event sequence, with 1722 regional consultations and 354 batched forward passes in total. Inferred maintenance averages 10.2 evaluations and 0.58 s, with 602 consultations and 207 batches. The measured evaluator-time reduction is $2.49\times$ before compilation. Charging one 0.56 s instruction compilation to each sequence gives 1.14 s of accounted model time versus 1.44 s for full recomputation, a 0.30 s saving. Savings depend on how many room judgments each event invalidates; compilation is amortized only when accumulated inference-time savings exceed its cost. The wall-clock figures are specific to one local model and one benchmark horizon. Timings are fresh inference with the tape disabled, wall clock around the generation calls on one L40S in bfloat16, one run without warm-up exclusion; the 0.56 s covers both $K=2$ compiler consultations. These figures exclude observation acquisition, change detection, planning, and motion.

Online footprints inferred from simulator state. The preceding experiments replay supplied footprints. In the closed-loop AI2-THOR [29] study, the controller instead obtains footprints by differencing consecutive simulator states. Qualification uses the exact benchmark predicate. Each episode applies five physical events in a fixed order (door closure, relocation, robot move, use, second relocation) and ends after the fifth; agreement is on the final target decision. Reachability is a breadth-first search over the live navigation grid, which a closed door splits, and *nearest* here is navigation-grid path length rather than doorway hops. This evaluates maintenance during execution with metadata-derived changes, not perception-driven grounding. Across 32 episodes spanning all six families, inferred maintenance matches the full-recomputation target decisions in all 32 episodes at 10.47 semantic evaluations versus 26.00, a $2.48\times$ reduction, with no stale support entries. Event-wise relevance matches the reference on 30/32 episodes at 11.97 evaluations.

Observation-derived removal footprints. OWLv2 [30] (google/owlv2-large-patch14-ensemble on 896×672 renders)

processes matched before/after room scans (Fig. 3) for known one-object removals, one standpoint per room at four headings, every room rescanned after the event from the same standpoints. Given the removed object type T , the candidate footprint includes rooms whose detection count for T changes and rooms whose prior count reaches the conservative multiplicity threshold. The operating point (detection threshold 0.05, multiplicity 6), fixed on 41 calibration events in 41 houses, contains the affected room for all 54 held-out houses, a disjoint index range of the same split, with a mean footprint of 2.19 of 4.61 rooms and a one-sided 95% Clopper–Pearson lower bound of 0.946. This evaluates affected-room localization given matched observations and a supplied event type, not event discovery; scan and detector cost is excluded.

E. Physical-robot demonstration

Fig. 4 shows MIND on a Unitree G1 humanoid in a four-zone laboratory: ROS 2 with lidar localization on a frozen map, Nav2 planning, a YOLO-World detector, and a 4-bit on-board judge (Qwen3-8B in scenarios 1 and 2, Phi-4 in 3 and 4). Signatures came from the deployed Phi-4 compiler, declared to the controller. Provided: zone polygons, candidate regions, and a scoring-only ground-truth inventory; observed: the robot’s region, planner reachability, and the detector’s inventory.

Four scenarios were run, 26 scored runs and 8 excluded with reasons. (1) *Something to sit on*, chairs in A and C, a backpack in the start room D: three evaluations selected A, and the walk through C, a transient loss and recovery of the route to A, and the arrival re-judged no room. MIND used 3 semantic evaluations and approximately 0.33 s of judge time per run over five runs, compared with 9 evaluations and approximately 1.0 s for full recomputation over three. (2) *The nearest thing to sit on*, same layout and move: the same move re-ranked the target from C to A without additional semantic evaluation under MIND, 3 against 9; in both scenarios all scored runs reached the expected target. (3) *A chair in the room I am in*, chairs in all three rooms: position is semantic, so every move re-judged every room under both policies and nothing was saved, as (6) predicts for a global refresh. The judge read the inventory line but not the position line, so every decision in the six runs inherited the same error while navigation and perception passed; the two policies agreed on every decision, as a fixed evaluator requires. (4) *Something to sit on*, then a barrier closing the route to A and the chair removed from C: the barrier refreshed reachability only and the removal re-judged one room, 4 evaluations against 15 to 18, and both policies abstained; one floor span was declared as wall so the barrier closed the only route. Evaluation counts per event were identical across repetitions; one full-recomputation run was excluded for a detector miss, a perception failure scored apart from the decision. The physical-robot demonstrations are also shown in the supplementary video.

VII. CONCLUSION

MIND combines instruction-conditioned qualification and selection dependencies with event footprints to reduce seman-

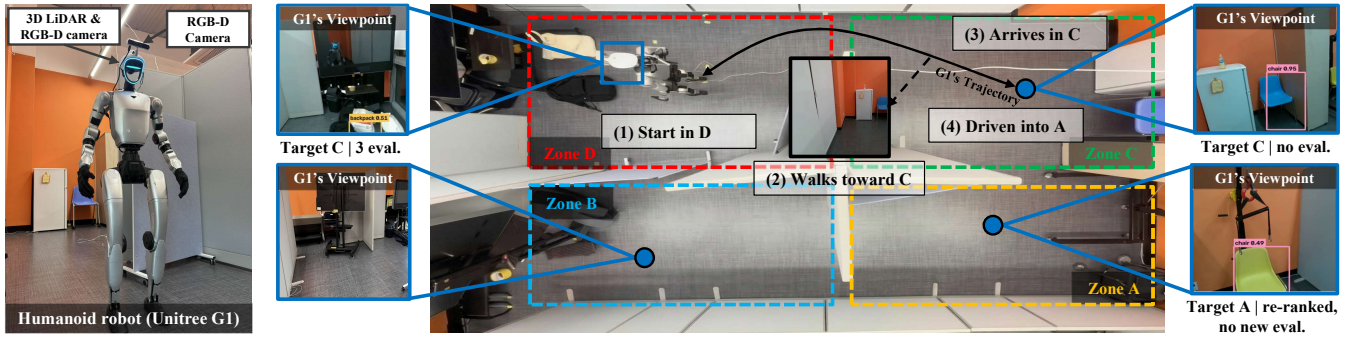


Fig. 4: **Unitree G1 hardware validation.** Left: the robot, with the 3D lidar and the RGB-D camera used for localization and detection. Centre: ceiling view of the four-zone laboratory, chairs in A and C and a backpack in the start zone D, with the path taken in one run of scenario 2, *the nearest thing to sit on*. Insets: the robot’s camera in each zone; the three moments of the run carry the target and the evaluation count from the module log. Three initial evaluations select C, one hop away, and the robot walks there and arrives. When the operator then drives it into A, the target re-ranks to A on position alone; position is a selection dependency for this request, so no room is re-judged.

tic re-evaluation. Under Proposition 1’s assumptions, selective maintenance preserves full-recomputation target decisions. The evaluations separate savings from spatial scope, request-specific variables, and stage placement, and expose the consequences of missed dependencies and footprints.

The controlled benchmarks assume populated semantic maps and a restricted state vocabulary; dependency labels are not independently annotated. Metadata-derived closed-loop updates and known-removal image localization test different components. Frozen verdict tapes isolate maintenance fidelity, not task correctness or online determinism. The hardware evidence is four scenarios on one layout. In the current-region case both policies inherited the same verdicts from an evaluator that misread the position field; semantic judgment is confined to one replaceable evaluator rather than special-cased in code, so a stronger judge changes the maintenance rule by nothing. Broader instruction coverage and robustness under changing evaluator outputs remain open.

ACKNOWLEDGMENT

Claude was used to assist with development and debugging of experimental software and data processing. All AI-assisted content was reviewed and validated by the authors.

This research was supported by the NVIDIA Academic Grant Program using NVIDIA RTX PRO 6000 Blackwell GPUs and by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2026S1A5A8012989)..

REFERENCES

- [1] U. A. Acar, G. E. Blelloch, and R. Harper, “Adaptive functional programming,” *ACM Transactions on Programming Languages and Systems*, vol. 28, no. 6, pp. 990–1034, 2006.
- [2] A. Gupta and I. S. Mumick, “Maintenance of materialized views: problems, techniques, and applications,” *IEEE Data Engineering Bulletin*, vol. 18, no. 2, pp. 3–18, 1995.
- [3] A. Mokhov, N. Mitchell, and S. Peyton Jones, “Build systems à la carte,” in *Proceedings of the ACM on Programming Languages (ICFP)*, vol. 2, 2018, pp. 79:1–79:29.
- [4] J. Doyle, “A truth maintenance system,” *Artificial Intelligence*, vol. 12, no. 3, pp. 231–272, 1979.
- [5] M. Ahn *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 205. PMLR, 2022, pp. 287–318, arXiv:2204.01691.
- [6] W. Huang *et al.*, “Grounded decoding: Guiding text generation with grounded models for embodied agents,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023, arXiv:2303.00855.
- [7] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, “Text2Motion: From natural language instructions to feasible plans,” *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023, special issue on Large Language Models in Robotics; arXiv:2303.12153.
- [8] W. Huang *et al.*, “Inner monologue: Embodied reasoning through planning with language models,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 205. PMLR, 2022, pp. 1769–1782, arXiv:2207.05608.
- [9] S. Yao *et al.*, “ReAct: Synergizing reasoning and acting in language models,” in *International Conference on Learning Representations (ICLR)*, 2023, arXiv:2210.03629.
- [10] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Sünderhauf, “SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning,” in *Proceedings of the 7th Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 23–72, oral presentation; arXiv:2307.06135.
- [11] A. Z. Ren *et al.*, “Robots that ask for help: Uncertainty alignment for large language model planners,” in *Proceedings of the 7th Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 661–682, arXiv:2307.01928.
- [12] J. Park *et al.*, “CLARA: Classifying and disambiguating user commands for reliable interactive robotic agents,” *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1059–1066, 2024, arXiv:2306.10376.
- [13] E. Musumeci *et al.*, “Context matters! relaxing goals with LLMs for feasible 3D scene planning,” *arXiv preprint arXiv:2506.15828*, 2025.
- [14] H. Yang, Z. Jiao, S. Wang, Y. Niu, S. Liu, and H. Liu, “Integrated exploration and sequential manipulation on scene graph with LLM-based situated replanning,” in *arXiv preprint*, 2026, arXiv:2602.04419.
- [15] G. Kim, T. Kim, S. S. Kannan, V. L. N. Venkatesh, D. Kim, and B.-C. Min, “DynaCon: Dynamic Robot Planner with Contextual Awareness via LLMs,” *arXiv preprint*, 2023, arXiv:2309.16031.
- [16] M. Lei *et al.*, “STMA: A Spatio-Temporal Memory Agent for Long-Horizon Embodied Task Planning,” *arXiv preprint*, 2025, arXiv:2502.10177.
- [17] H. Xu, Z. He, H. Wang, J. Xu, and H. Dong, “Mimir: A neuro-symbolic memory system with dynamic grounding for embodied agents in interactive environments,” in *arXiv preprint*, 2026, arXiv:2608.04933.
- [18] P. Nguyen, F. Verdoja, and V. Kyrki, “Event-grounding graph: Unified spatio-temporal scene graph from robotic observations,” *IEEE Robotics and Automation Letters*, vol. 11, no. 5, pp. 5286–5293, 2026.
- [19] D. Kim, N. Oh, D. Hwang, and D. Park, “LINGO-Space: Language-conditioned incremental grounding for space,” in *Proc. AAAI Conf. Artificial Intelligence*, 2024.
- [20] Y. Sun and Y. Zhang, “When memory lies: an empirical study of spatial memory staleness in VLM agents,” *arXiv preprint*, 2026, arXiv:2608.04574.
- [21] Z. Yang, T. Xie, B. Lu, S. Liu, B. Yu, and M. Li, “KEEP: A KV-cache-centric memory management system for efficient embodied planning,” in *arXiv preprint*, 2026, arXiv:2602.23592.
- [22] H. Kim, Y. Wu, and T. Tambe, “AgenticCache: Cache-driven asynchronous planning for embodied AI agents,” in *Proceedings of Machine Learning and Systems (MLSys)*, 2026, code at github.com/hojoonleokim/MLSys26_AgenticCache.
- [23] K. Obata, T. Aoki, T. Horii, T. Taniguchi, and T. Nagai, “LiP-LLM: Integrating linear programming and dependency graph with large language models for multi-robot task planning,” in *arXiv preprint*, 2024, arXiv:2410.21040.
- [24] P. Pramanick, H. B. Barua, and C. Sarkar, “DeComplex: Task planning from complex natural instructions by a collocating robot,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, arXiv:2008.10084.
- [25] Y. Wang *et al.*, “DART-LLM: Dependency-Aware Multi-Robot Task Decom-

- position and Execution using Large Language Models,” *arXiv preprint*, 2024, arXiv:2411.09022.
- [26] M. Abrams, T. Oelerich, and M. Scheutz, “LLM-Augmented Incremental Language Parsing for Robotic Agents,” in *Proceedings of the Twelfth Annual Conference on Advances in Cognitive Systems*, 2025, pp. 194–209.
 - [27] S. Patki, J. Arkin, N. Raicevic, and T. M. Howard, “Language guided temporally adaptive perception for efficient natural language grounding in cluttered dynamic worlds,” in *IROS*, 2023.
 - [28] M. Deitke *et al.*, “ProcTHOR: Large-scale embodied AI using procedural generation,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 5982–5994, arXiv:2206.06994.
 - [29] E. Kolve *et al.*, “AI2-THOR: An interactive 3D environment for visual AI,” *arXiv preprint arXiv:1712.05474*, 2017.
 - [30] M. Minderer, A. Gritsenko, and N. Houlsby, “Scaling open-vocabulary object detection,” in *Advances in Neural Information Processing Systems*, vol. 36, 2023.