

HelpMeOut: A Neuropsychology-Grounded Benchmark for Error-Aware Monitoring of Household Task Execution

Haoran Wang¹, Jingwei Huang², Anqing Duan², Ho Seok Ahn¹,
Bruce Macdonald^{†,1}

¹University of Auckland, New Zealand

²MBZUAI, United Arab Emirates

[†]b.macdonald@auckland.ac.nz

Abstract: We introduce HELPMEOU, an embodied AI benchmark for error-aware monitoring of household task execution. Existing embodied household benchmarks typically assume that humans execute tasks correctly, but this assumption does not hold for cognitively impaired persons or older adults whose everyday activities produce structured execution errors. To address this gap, HELPMEOU operationalizes research on everyday action errors into 20,000 paired activity episodes, covering nine neuropsychologically motivated error categories. In each episode, agents observe a streaming human activity trajectory and must monitor whether the execution is unfolding correctly, diagnosing any structured errors as early as possible. We evaluate learned-monitor, symbolic, change-point, and frontier LLM prompting baselines. Results show that these methods perform poorly as error monitors, often missing when execution goes wrong. These results suggest a gap left by standard goal-inference and task-completion benchmarks, making HELPMEOU a testbed for error-aware assistive agents.

Keywords: Error-Aware Monitoring, Assistive Robotics, Embodied AI Benchmark

1 Introduction

Consider a cognitively impaired person making a cup of instant coffee. She adds the powder and pours hot water, but never stirs—the stirring step does not come to mind. Later, she takes a fork to add sugar, unaware that a spoon is the appropriate tool. These look like minor lapses, but to an assistive robot they pose a distinct problem: the person is not confused about her goal, nor is she asking for help—she is executing an everyday task under degraded cognitive control. How can the robot recognize, from partial household actions, that execution is going wrong?

Neuropsychological studies of everyday action show that such breakdowns are not random noise: everyday tasks often fail in structured ways such as omissions, substitutions, sequence errors, and perseverations [1, 2]. Detecting such errors and intervening at the right time is a core requirement for assistive robots supporting people with cognitive impairment and older adults. This is a large and growing population: cognitive impairments such as dementia affect tens of millions worldwide—dementia alone reaches 55 million people and is projected to grow to 139 million by 2050 [3, 4], and a further 12–18% of adults over 60 live with mild cognitive impairment, a condition often associated with deficits in instrumental activities of daily living [5]. Yet this capability is rarely evaluated in existing embodied household benchmarks.

Most such benchmarks share a common assumption: humans act coherently toward well-formed goals, and the agent’s central problem is inferring *what* the human wants to do. For cognitively impaired persons and older adults, however, the harder question is not only what the person is trying

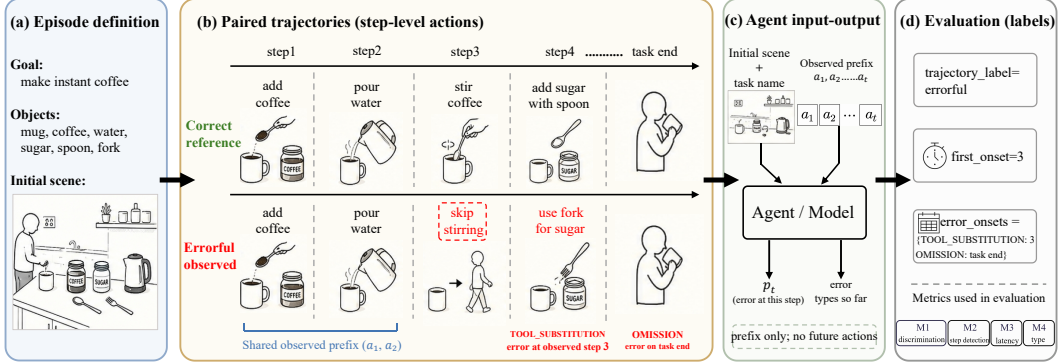


Figure 1: HELPMEOOUT task overview, shown with an instant-coffee task. **a.** We define the episode with goal, available objects, and initial scene. **b.** An episode consists of a pair of trajectories: the correct reference on top and the errorful observed execution on bottom, which contains a `TOOL_SUBSTITUTION` (fork used instead of spoon) and an `OMISSION` (stirring skipped), with the latter confirmed only when the task ends. **c.** The system input is the task name, the initial scene, and the action prefix in symbolic PDDL notation; the output is an error probability, onset step, and error type. **d.** Evaluation labels and the four metrics (M1–M4) derived from the error injection metadata; we explain them in detail in Sec. 3.3 and Sec. 3.4.

to do, but *whether* their ongoing execution is failing. A separate line of smart-home and robot-assisted systems does target execution failures directly [6, 7, 8], but typically relies on fixed sensor setups or predefined activity scripts, yielding task-specific signals rather than standardized, large-scale data for agent evaluation. What is missing is a scalable benchmark that tests step-level error monitoring from partial household activity streams. Figure 1 illustrates this setup with an instant-coffee episode.

We therefore introduce HELPMEOOUT, an embodied AI benchmark for error-aware monitoring of household task execution. Guided by the neuropsychological evidence above, HELPMEOOUT grounds its episodes in a goal-control model [9] whose parameters are calibrated against the per-category error rates reported in naturalistic-action studies of control and clinical populations [2, 10]: error type and frequency vary by error category and degree of cognitive impairment, rather than being sampled uniformly at random. Agents must therefore recognize errors whose structure reflects impaired everyday action, rather than artifacts of the sampling procedure. As in these tests, the task is known; the monitor must judge whether its execution goes wrong.

HELMEOOUT makes three contributions. First, we formalise online error-aware monitoring as a task of deciding whether, when, and what kind of execution error occurs while a household task unfolds. Second, we build a generator that turns Activities of Daily Living (ADL)/Instrumental Activities of Daily Living (IADL) tasks into paired correct and errorful episodes, driven by a goal-control model from the neuropsychology of naturalistic action, and release a dataset of 20,000 paired episodes¹. Third, we cast the task as a streaming evaluation protocol that separately measures error discrimination, step-level detection, latency, and type diagnosis. Experiments show that learned baselines struggle with step-level error detection and diagnosis, symbolic baselines with additional domain knowledge primarily identify incomplete tasks at episode termination without distinguishing error types, and frontier LLMs remain limited in step-level error localization and timely alerting.

2 Related Works

Everyday action errors. In the past few decades, clinical neuropsychology studied everyday action impairment using performance-based instruments such as the Naturalistic Action Test (NAT), the Multi-Level Action Test (MLAT), and the Performance Assessment of Self-Care Skills

¹Code, data, and the symbolic simulator: <https://whrcn.github.io/HelpMeOut-Official/>

Benchmark	Long-horiz.	Object interact.	Subgoal ctx.	Errorful human exec.	Neuropsych err. taxonomy	Per-step err. labels	Latency eval.	Streaming eval.
TextWorld [18]	✓	✓	✗	✗	✗	✗	✗	△
ALFRED [19]	△	✓	✗	✗	✗	✗	✗	✗
VirtualHome [20]	✓	✓	✗	✗	✗	✗	✗	✗
Watch-and-Help [21]	✓	✓	✓	✗	✗	✗	✗	✗
BEHAVIOR / 1K [22, 23]	✓	✓	✗	✗	✗	✗	✗	✗
HandMeThat [24]	✓	✓	✓	✗	✗	✗	✗	✗
BALI [25]	✓	✓	✓	✗	✗	✗	✗	✗
HELPMEOU (ours)	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison with related benchmarks. ✓ present; ✗ absent; △ partial. Existing benchmarks evaluate goal inference or task completion under coherent human execution. HELPMEOU introduces errorful execution, a neuropsychological error taxonomy, step-level error labels, detection-latency evaluation, and streaming evaluation.

(PASS) [1, 2, 11, 10, 12]. These tests ask participants to perform familiar multi-object tasks and score errors such as omissions, sequence errors, substitutions, additions, and perseverations. The goal-control account [9] draws on the Supervisory Attentional System (SAS) framework [13] and contention-scheduling models [14, 15], linking omissions to weak or decaying task goals and commission errors to failures to suppress competing goal activations, especially around subgoal transitions [16, 17]. An alternative connectionist account models the same error categories through recurrent dynamics without explicit schema hierarchies [26]. More recently, these error taxonomies have been extended to characterise subtle, sub-clinical errors in mild cognitive impairment [27].

Assistive error detection systems. Smart-home and robot-assisted systems have used sensors or predefined activity scripts to detect deviations and prompting opportunities [6, 7, 8]. Deployed prompting systems include COACH, which uses a POMDP over video observations to guide dementia patients [28, 29], and SmartPrompt2, which targets common omission errors [30]. Related datasets capture erroneous kitchen and procedural activities: CaptainCook4D records ego-centric cooking videos in which participants deviate from recipes to induce errors [31], while the Kitchen Task Assessment dataset records sensor data from participants simulating typical dementia errors [32]. Existing methods also address procedural error detection: PREGO compares recognized steps against LLM-predicted steps [33], differentiable task-graph learning checks steps against a learned task graph [34], and MistExit aims to identify erroneous pre-segmented key steps as early as possible [35]. These works demonstrate the importance of detecting execution failures, but are often constrained by specific tasks, sensor setups, or errors that participants induce or simulate. As a result, they do not provide a scalable benchmark for evaluating whether models can detect neuropsychologically grounded error types as they unfold, with explicit onset labels and first-alarm timing metrics. Large-scale data collection from cognitively impaired users is also constrained by privacy and ethical concerns. HELPMEOU addresses these limitations by converting the neuropsychological taxonomy and goal-control account into automatically labelled episodes, with clinically calibrated error rates and deterministic step-level onsets, for streaming agent evaluation.

Embodied household benchmarks and simulators. Monitoring an ongoing task is closely related to goal and plan recognition from partial observations [36, 37]. Classical methods usually assume that the observed agent acts rationally; bounded-rational variants relax this assumption by allowing sub-optimal plans [38], which is closer to the impaired executions considered in HELPMEOU; such work infers goals from imperfect behavior, whereas HELPMEOU asks whether, when, and how the execution itself goes wrong. LLMs are also increasingly used as embodied household planners [39, 40], motivating their evaluation as error monitors in our benchmark.

Two adjacent lines of work share HELPMEOU’s household setting. The first evaluates agents that infer human intent or resolve instructions from observed behavior and language, including HandMeThat [24], FISER [41], BALI [25], and Watch-and-Help [21]. The second provides scalable household task simulators and trajectory generators, including TextWorld [18], BEHAVIOR

and BEHAVIOR-1K [22, 23], VirtualHome [20], ALFRED [19], and Habitat rearrangement [42]. Across both lines, however, the human action stream is assumed competent and the resulting trajectories are correct by construction. HELPMEOU keeps the scalable household-generation setting but adds neuropsychologically grounded errorful trajectories, paired correct references, and step-level onset labels for error-aware monitoring (Table 1).

3 The HELPMEOU Benchmark

3.1 Preliminaries: Neuropsychology of Everyday Action

HELMEOU uses the neuropsychological literature on everyday action impairment to define both its error labels and its generation model. Performance-based instruments such as NAT, MLAT, and PASS identify recurring failures in familiar multi-object tasks, including omissions, sequence errors, substitutions, additions, and perseverations [1, 2, 11, 10, 12]. These categories recur across clinical populations such as CVA, traumatic brain injury, and dementia, and provide the basis for our benchmark labels.

For generation, we adopt the goal-control account [9], which separates failures of goal activation from failures of executive control. In our benchmark, goal-activation failures produce omissions, while executive-control failures produce commission errors such as substitutions, additions, and captures. This abstraction gives the generator two interpretable control factors: whether a required subgoal is omitted, and whether a competing action is inserted or substituted. Sec. 3.3 describes how these factors are instantiated in the error generator. Figure 2 summarizes this generation process. Additional background on the neuropsychological literature is provided in the supplementary material.

3.2 Basic Monitoring Setting

HELMEOU is built over a symbolic household domain with 156 object types and 38 grounded action operators covering object manipulation, tool use, and state transformations. At evaluation step t , an agent observes the task name, the initial symbolic scene representation, and the action prefix so far, including the current action. The task is known to the monitor, as in the naturalistic-action tests on which our error categories are based: in the NAT and MLAT, the examiner names the task and its requirements before the person begins, and errors are scored against those requirements [11, 10]. Unlike the examiner, the monitor receives only the task name, not its formal requirements, and must infer what the task calls for. From this prefix, the agent must answer three questions: whether execution has already gone wrong, when the observable error onset occurred, and which error categories have appeared.

Each HELPMEOU episode consists of paired trajectories. A symbolic planner first generates a correct household execution from an ADL/IADL task template, and the error process then modifies that execution to obtain an errorful counterpart. The templates cover familiar multi-object routines such as food preparation, packing, cleaning, laundry, medication, and personal routines, following task families used in clinical and functional assessments of everyday action [10, 12, 43, 44, 45]. Formally, we write each episode as

$$e = \langle s_0, \Omega_{\text{correct}}, \Omega_{\text{err}}, S_0, y_{0:T'-1}, O \rangle. \quad (1)$$

Here, s_0 is the initial household state, Ω_{correct} is the reference execution, and Ω_{err} is the realized errorful execution of length T' . S_0 is the initial symbolic scene representation exposed to agents, encoding object identities, locations, states, affordances, and spatial relations. The sequence $y_{0:T'-1}$ gives sparse onset annotations: a step is labelled with `has_error`= 1 only when an injected error first becomes observable. $O = \{(\text{type}_j, t_j)\}_j$ records each error category and onset step, which supports step-level error detection, first-alarm timing, and cumulative error-type diagnosis. Subgoal boundaries and generation goals are retained as dataset metadata for constructing and analysing episodes; they are not shown to agents. Every episode ends with an explicit task-end event, at which the person declares the whole task finished.

Structured Error Injection from Clinical Profiles

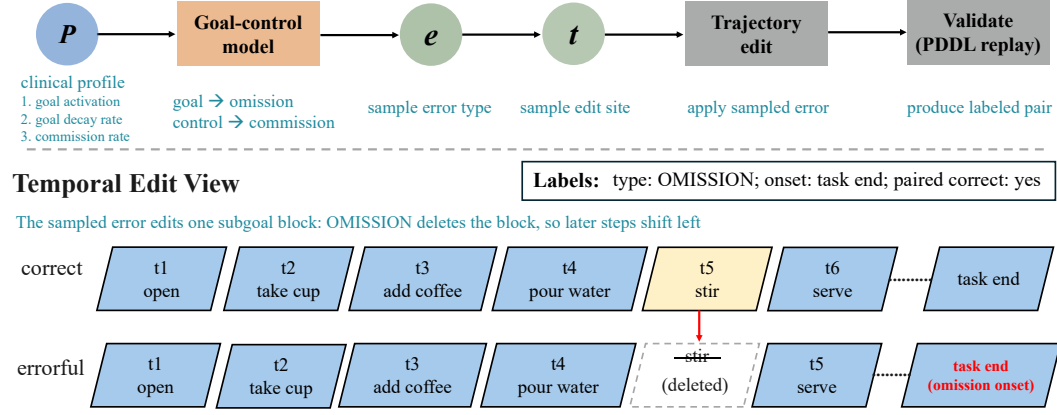


Figure 2: Neuropsychology-based error-generation pipeline. A neuropsychological profile drives the goal-control model to sample an error type and the subgoal block to edit; the edited trajectory is validated by symbolic replay, and onsets are derived by category-specific rules. The temporal view shows an OMISSION deleting one subgoal block and shifting later steps left; the omission is confirmed at the task-end event.

An error’s observable onset is the earliest step at which it can be identified from the observed execution prefix and task constraints, using category-specific rules: the first use of a substitute object or tool, an out-of-order action, an action repetition, committing an unprocessed item, placing an unrelated item in the task container, acting on an unrelated object, or leaving an unrelated storage location empty-handed. An omission removes an entire subgoal block. Since valid subgoal orders vary, a missing subgoal may still be completed later; thus, omission becomes observable only at the task-end event. An earlier consequence, such as storing an unprocessed item, is instead labelled as anticipation under MLAT coding [11]. Computed automatically from s_0 , Ω_{err} , and the injected category, onset annotations are deterministic and reproducible without human raters.

3.3 Structured Execution Errors

Goal-Control error process. Inspired by the goal-control account [9], we treat omissions as goal-activation failures and commission errors as executive-control failures. We operationalise this distinction by computing, for each subgoal block b_k , an activation level:

$$u_k = \theta_{goal}(1 - \theta_{decay})^k + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \sigma^2), \quad (2)$$

where θ_{goal} controls baseline goal salience and θ_{decay} captures accumulating cognitive load across subgoals. If $u_k < \tau_{omit}$, the subgoal is omitted; otherwise, with probability θ_{ctrl} , a commission error is sampled from a task-conditioned distribution modulated by per-category feasibility. We calibrate θ_{goal} , θ_{decay} , θ_{ctrl} and the category weights w_c so that the marginal error distribution produced by the generator matches the per-category error rates reported for control and clinical populations by Schwartz et al. [10] and Giovannetti et al. [2]. The resulting impaired profile ($\theta_{goal}=0.48$, $\theta_{decay}=0.20$, $\theta_{ctrl}=0.12$) aggregates rates across left-CVA, right-CVA, TBI, and dementia subjects in those two studies. Numerical values of τ_{omit} and the per-category feasibility scores $\phi_c(s_k)$ are reported in the supplementary material.

Error taxonomy. The nine error categories are grounded in MLAT/NAT taxonomies summarized by the goal-control model [9]. We organise them into two levels: progression-level errors that disrupt the structure of task execution (OMISSION, SEQUENCE, ANTICIPATION, PERSEVERATION), and current-action errors that primarily affect local action selection, object use, or spatial targeting (OBJECT_SUBSTITUTION, TOOL_SUBSTITUTION, SPATIAL_MISREACH, ACTION_ADDITION, CAPTURE). Detailed definitions and examples for each error category are provided in the supplementary material. This distinction matters for assistance: progression-level errors require reactivation or

redirection of the task plan, whereas current-action errors require immediate correction of the ongoing action. Consistent with clinical reports across neurological populations, omissions are treated as a prominent error class, while perseverations are sampled sparsely to reflect their lower base rate. Correct executions vary in subgoal order and intermediate placements, as the generator randomizes subgoal order and plans each subgoal separately. A reordering is labelled SEQUENCE only if it violates a required order (stirring before all ingredients are added, or a step-order norm such as buttering bread before toasting); otherwise, it remains correct.

Dual-standard validation. A key challenge is that errorful trajectories must be semantically wrong yet physically executable: a naive PDDL validator would reject substitution errors, because using a fork where a spoon is required is the violation, while an unconstrained validator admits physically impossible traces. We resolve this by applying strict precondition replay to the seven non-substitution error types, and for substitution errors enforcing only the physical chain—navigation, accessibility, containment, hand occupancy—while relaxing the single semantic precondition that defines the substitution. This ensures that injected errors remain interpretable as human execution errors rather than simulator failures: every errorful trajectory is physically executable, and the person can carry out each of its actions.

Dataset statistics. The HELPMEOU benchmark spans 50 ADL/IADL task templates and contains 20,000 episodes—12,000 errorful and 8,000 correct, with correct episodes paired with themselves and labeled error-free. Errorful trajectories average 24.7 steps, including the task-end event, with 1.07 injected errors each (6.8% have more than one), yielding 12,746 annotated onsets; omissions are the most frequent category (5,346), followed by sequence errors (1,478) and object substitutions (1,142). We hold out 500 test episodes (312 errorful, 188 correct) and a 2,022-episode validation split; the remaining 17,478 episodes are for training.

3.4 Prefix-Based Evaluation

We evaluate online error monitoring as a streaming task: at step t , the agent observes the task name, S_0 , and the action prefix $a_{0:t}$, and outputs p_t , the probability that an error occurs at step t , together with cumulative probabilities for the nine categories; no future action is visible. For a trajectory with N actions, the first $\lfloor N/4 \rfloor$ steps are a context window that is not scored, and all onsets fall in the remaining evaluation window, which ends with the task-end event. The four metrics correspond to the benchmark questions of whether something went wrong, where the error becomes observable, when the model raises its first alarm, and what type of error occurred:

M1—Trajectory-level discrimination. The score of a trajectory is $s = \max_t p_t$ over the evaluation window, and AUROC measures whether errorful trajectories score above correct ones. M1 does not depend on when the alarm is raised; timeliness is measured by M2 and M3. **M2—Step-level detection.** Each evaluated step is scored against its label y_t with AUPRC. Only an onset step is positive; later steps, where the error’s consequences persist, are negative, so a late alarm earns no credit and counts as a false alarm. Because omissions are confirmed at the task-end step, we also report M2-end, the AUPRC at that step, and M2-mid, the AUPRC at the earlier steps where all other errors occur. **M3—First-alarm timing.** For the first-alarm step $\hat{t} = \min\{t : p_t \geq \theta\}$ with $\theta = 0.5$, latency is $\Delta = \hat{t} - t_{\text{onset}}$. We group alarms as early ($\Delta < 0$), on target ($\Delta = 0$), late ($\Delta > 0$), or missed, and report the false-alarm rate on correct episodes. Omissions, which can only be confirmed at task end, are reported separately. **M4—Error-type diagnosis.** The cumulative category probabilities are scored against the set of categories that have occurred so far (multi-label): in errorful trajectories from the first onset onward, and in correct trajectories at every evaluated step, where this set is empty and naming any category is a false positive. Each trajectory is weighted equally, and we report macro-F1. A monitor is thus credited for naming an error’s category only where an error has actually occurred.

Table 2: Main results on the 500-episode test set (metrics in Sec. 3.4). M3 at $\theta=0.5$ over the 175 episodes with an error before task end; FA over the 188 correct episodes (re-planner: timed-out episodes excluded). Random: positive-step rate for M1 and M2, a uniformly random monitor for M3 and M4 (mean of 20 seeds). [†]Human reference on the 60-episode subset (M2 random 0.027; M3 over 26 errorful and 30 correct episodes). Symbolic references use extra knowledge (Sec. 4).

Method	M1	M2			M3				M4
		all	end	mid	Missed	Early	On target	FA	
Random	0.500	0.036	0.274	0.023	0.00	0.89	0.05	1.00	0.116
Human [†]	0.967	0.748	–	–	0.12	0.15	0.73	0.17	0.778
<i>Learned and change-point monitors</i>									
Seq2Seq	0.496	0.376	0.656	0.132	0.99	0.00	0.00	0.10	0.502
Seq2Seq+SG	0.554	0.394	0.665	0.151	0.98	0.00	0.00	0.05	0.548
SGT	0.778	0.656	0.791	0.553	0.52	0.04	0.38	0.12	0.542
CUSUM	0.645	0.039	0.235	0.029	0.06	0.77	0.07	0.81	0.000
<i>Symbolic references (extra knowledge)</i>									
Precondition check	0.513	0.059	0.274	0.062	0.95	0.00	0.05	0.00	0.034
Re-planner	0.887	0.379	0.690	0.118	0.10	0.75	0.05	0.62	0.138

Table 3: SGT by error category on the test set. Each category is scored against all correct episodes: M1 over episodes, M2 over steps (for omissions, all evaluated steps with the task-end step as the positive; for other categories, the steps before task end), with Rand. its positive-step rate. On target and missed ($\theta=0.5$) are shares of the category’s episodes; M4 is per-class F1.

Category	n	M1	M2	Rand.	On target	Missed	M4 F1
Omission	137	0.831	0.730	0.026	58%	41%	0.916
Sequence	38	0.891	0.814	0.010	74%	26%	0.921
Anticipation	17	0.861	0.359	0.005	29%	47%	0.607
Perseveration	26	0.725	0.502	0.007	42%	54%	0.803
Object substitution	30	0.522	0.038	0.008	0%	87%	0.196
Tool substitution	9	0.501	0.020	0.003	0%	89%	0.000
Action addition	9	0.405	0.029	0.003	11%	89%	0.216
Capture	23	0.686	0.178	0.012	13%	70%	0.502
Spatial misreach	23	0.952	0.771	0.007	83%	4%	0.715

4 Experimental Results

Evaluation protocol. We evaluate three families of monitors. *Learned*: Seq2Seq, which encodes the initial scene and the observed action prefix; Seq2Seq+SG, which additionally encodes the symbolic scene graph replayed from the PDDL state (object identity, location, state, and containment; no geometry); and a scene-graph transformer (SGT) that reads the replayed graph before each action together with the action itself. *Non-learned*: a CUSUM change-point detector over action-bigram surprise fitted on correct training trajectories; a precondition checker that flags actions whose PDDL preconditions fail; and a re-planner that, given the task’s template-level goal, plans with the generator’s planner, tracks the plan, re-plans when the observed action deviates, and raises an alarm when the re-plan grows longer, when an action cannot execute, or when the goal is unmet at task end. *LLMs*: frontier models prompted step by step with the same prefix-only input. All learned and non-learned baselines run on the full 500-episode test set; because LLM evaluation is costly, LLMs and the Human reference use a 60-episode subset (30 errorful, 30 correct). For the Human reference, annotators saw the task setting (objects, initial scene, and error categories) and judged each prefix under the same whether, when, and what-kind questions. We report 95% confidence intervals from a trajectory-level bootstrap stratified by errorful and correct episodes, and we do not rank methods whose intervals overlap. Implementation details are in the supplementary material.

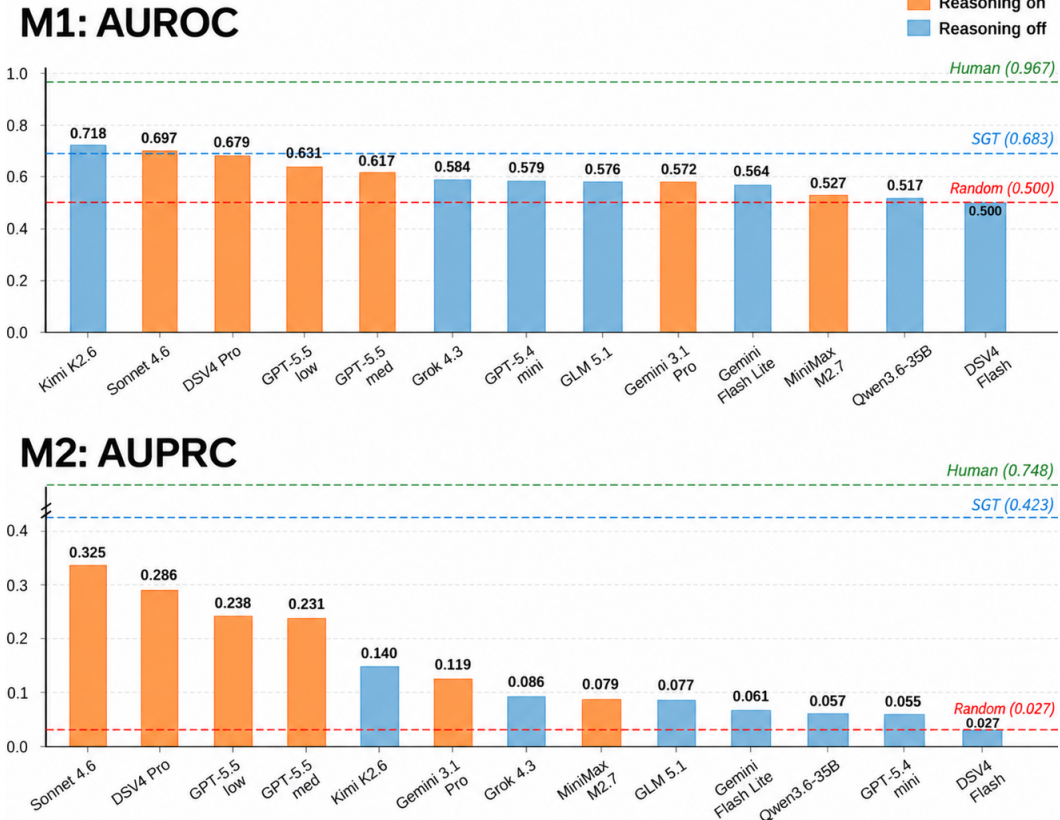


Figure 3: M1 AUROC and M2 AUPRC of the thirteen LLM configurations on the 60-episode subset; dashed lines mark the Human, SGT, and Random references, and color marks whether reasoning is enabled. Confidence intervals and full M1–M4 results are in the supplementary material.

Learned and change-point baselines. Table 2 summarizes the results. Seq2Seq and Seq2Seq+SG detect error steps well above chance on M2 (0.376 and 0.394 vs. 0.036), but most of this comes from the task-end step, where omissions are confirmed (end-step AUPRC 0.66 vs. 0.13–0.15 at the remaining steps), and their trajectory-level discrimination stays near chance (M1 0.496 and 0.554). Seq2Seq+SG improves over Seq2Seq on M1 (+0.058, 95% CI [+0.023, +0.092]) and M4 (+0.047, [+0.019, +0.077]), but not significantly on M2. SGT, which conditions every step on the current action, is the strongest learned monitor (M1 0.778, M2 0.656; 0.553 at non-final steps); its gains concentrate on errors with local action signatures (sequence, spatial misreach, perseveration), while substitutions and action additions remain near chance (Table 3). CUSUM reaches M1 0.645 only by alarming on 81% of correct episodes, with M2 at chance and no type diagnosis, which is why M1 alone cannot measure monitoring. All methods remain far below Human (M1 0.967, M2 0.748).

Symbolic baselines. Unlike the learned monitors, the symbolic baselines use the PDDL domain model, and the re-planner additionally receives the task’s formal goal and the generator’s planner; they are knowledge-rich references rather than competitors, and they test whether the benchmark’s errors can be caught by construction. Because seven of the nine categories are executable under strict PDDL replay, the precondition checker fires on fewer than 5% of errorful episodes (M1 0.513). The re-planner attains the highest M1 (0.887), but almost entirely through its end-of-task check: once the person declares the task finished, an unmet goal confirms an omission with no false alarms, and this check alone gives M1 0.802. Restricted to steps before task end, its M1 drops to 0.669 [0.622, 0.714], overlapping with the other baselines (0.59–0.62); its step-level AUPRC at non-final steps is 0.118 (SGT: 0.553); and 62% of correct episodes receive an alarm, because valid human executions are not shortest plans and deviation from a plan is not an error. A symbolic monitor can

thus confirm after the fact that a task was left incomplete, but it does not detect or localize errors while execution unfolds.

Alarm timing and type diagnosis. At $\theta=0.5$, the Seq2Seq models almost never raise an alarm (miss rate 0.99 and 0.98): their per-step scores rank error steps above other steps but rarely cross the threshold. SGT is the only monitor whose alarms are mostly timely—80% of its first alarms fall at the onset step, at a 12% false-alarm rate on correct episodes—but it misses 52% of errorful episodes, so only 38% receive an on-target alarm; the misses concentrate in the substitution, action-addition, and capture categories (70–89% missed), whereas 83% of spatial misreaches receive an on-target alarm (Table 3). CUSUM and the re-planner rarely miss, but their first alarm comes early in about three quarters of errorful episodes, and they alarm on 81% and 62% of correct episodes. On M4, the learned monitors reach macro-F1 0.50–0.55 (per-class F1 in Table 3 and the supplementary material). Their type heads also name an error category in most correct episodes (all 188 for both Seq2Seq models, 85% for SGT), which M4 penalizes; scored on errorful episodes alone, their macro-F1 would be 0.63–0.68. The Seq2Seq models thus name error categories confidently while almost never raising an alarm, so their type and alarm outputs are not consistent with each other.

LLM baselines do not close the monitoring gap. Figure 3 compares thirteen LLM configurations on the 60-episode subset. The upper ends of the best LLM confidence intervals stay below Human on both M1 (0.837 vs. 0.967) and M2 (0.482 vs. 0.748), and on M2 no LLM exceeds SGT on the same episodes (0.423). Intervals overlap substantially across models, so we do not rank them. Reasoning matters most for localization and diagnosis: with reasoning enabled, the larger models (Claude Sonnet 4.6, DeepSeek V4 Pro, GPT-5.5, Gemini 3.1 Pro) reach M2 0.12–0.33 and M4 0.43–0.52, whereas with reasoning disabled M2 stays at or below 0.14 and M4 at or below 0.31. More reasoning effort does not help further (GPT-5.5 low vs. medium: M2 0.238 vs. 0.231). With reasoning enabled, these models also give the most on-target first alarms (42–50% of errorful episodes, vs. 23% for SGT on the same subset), but they alarm on 30–77% of correct episodes. Without reasoning, several models output near-zero scores at almost every step. LLMs can thus often recognize that something is off, but they still struggle to identify the step at which an error occurs and to raise a timely alarm, whereas the human reference raises an on-target alarm in 73% of errorful episodes at a 17% false-alarm rate.

5 Limitations and Broader Impact

HELPMEOU is symbolic: episodes are discrete action sequences over a PDDL-style state, which gives large-scale data with deterministic onsets but assumes that actions have already been recognized from video or sensors. It therefore captures neither recognition errors nor the multimodal cues through which errors manifest (hesitation, gaze, fumbling), and its results are an upper-bound proxy for symbolic-level monitoring; grounding the episodes in video or real ADL recordings is left to future work.

Two further limits follow from how episodes are generated. First, every errorful trajectory is physically executable, so failures in which an earlier lapse makes a later action impossible, such as pouring from a kettle that was never filled, are not represented. Second, correct executions are produced by a planner that completes one subgoal before starting the next; they vary in order and intermediate placements but do not interleave subgoals or contain the harmless redundancy of everyday human action, and insertion-type errors are therefore partly identifiable from surface patterns. The errors are generated rather than recorded: a label-blinded review by two clinical experts judged most generated errors plausible (supplementary material), but this is not patient-level validation.

Deployment raises risks the benchmark does not measure. False alarms interrupt the person and erode their autonomy and trust, whereas missed alarms can let an unsafe lapse, such as a stove left on, go unnoticed; thresholds should therefore be set with users and caregivers rather than by a benchmark score. In-home monitoring also requires informed consent, including from people whose capacity to consent may change, data minimization, and the person’s control over what is

recorded. HELPMEOU contains only synthetic episodes and no personal data, and results on it are not evidence that a system is safe to deploy.

6 Conclusions

We presented HELPMEOU, a benchmark that reframes household assistance as online error monitoring rather than goal inference. By pairing correct executions with neuropsychologically grounded errorful ones and asking whether, when, and what kind of error occurs, it exposes a gap that goal-inference and task-completion benchmarks miss: learned monitors, symbolic references, and frontier LLMs can often tell that an episode went wrong, especially once it has ended, but rarely detect and localize errors as they happen, and errors that require knowing which object or tool the task calls for remain near chance. We hope HELPMEOU serves as a testbed for assistive agents that catch execution breakdowns before a task fails. Results of further expert validation will be released on the project website.

References

- [1] L. J. Buxbaum. Ideational apraxia and naturalistic action. *Cognitive Neuropsychology*, 15 (6-8):617–643, 1998.
- [2] T. Giovannetti, D. J. Libon, L. J. Buxbaum, and M. F. Schwartz. Naturalistic action impairments in dementia. *Neuropsychologia*, 40(8):1220–1232, 2002.
- [3] World Health Organization. Dementia. Fact sheet, 2025. URL <https://www.who.int/news-room/fact-sheets/detail/dementia>. Accessed: 2026-04-29.
- [4] E. Nichols, J. D. Steinmetz, S. E. Vollset, K. Fukutaki, J. Chalek, F. Abd-Allah, A. Abdoli, A. Abualhasan, E. Abu-Gharbieh, T. T. Akram, et al. Estimation of the global prevalence of dementia in 2019 and forecasted prevalence in 2050: an analysis for the global burden of disease study 2019. *The Lancet Public Health*, 7(2):e105–e125, 2022.
- [5] S. A. Olatunji, J. S. Shim, A. Syed, Y.-L. Tsai, A. E. Pereira, H. P. Mahajan, R. A. Mudar, and W. A. Rogers. Robotic support for older adults with cognitive and mobility impairments. *Frontiers in Robotics and AI*, 12:1545733, 2025.
- [6] B. Das, D. J. Cook, N. C. Krishnan, and M. Schmitter-Edgecombe. One-class classification-based real-time activity error detection in smart homes. *IEEE journal of selected topics in signal processing*, 10(5):914–923, 2016.
- [7] G. Wilson, C. Pereyda, N. Raghunath, G. De La Cruz, S. Goel, S. Nesaei, B. Minor, M. Schmitter-Edgecombe, M. E. Taylor, and D. J. Cook. Robot-enabled support of daily activities in smart home environments. *Cognitive systems research*, 54:258–272, 2019.
- [8] C. Pereyda, N. Raghunath, B. Minor, G. Wilson, M. Schmitter-Edgecombe, and D. J. Cook. Cyber-physical support of daily activities: A robot/smart home partnership. *ACM Transactions on Cyber-Physical Systems*, 4(2):1–24, 2019.
- [9] T. Giovannetti, R. Mis, K. Hackett, S. M. Simone, and M. B. Ungrady. The goal-control model: An integrated neuropsychological framework to explain impaired performance of everyday activities. *Neuropsychology*, 35(1):3, 2021.
- [10] M. F. Schwartz, M. Segal, T. Veramonti, M. Ferraro, and L. J. Buxbaum. The naturalistic action test: A standardised assessment for everyday action impairment. *Neuropsychological rehabilitation*, 12(4):311–339, 2002.
- [11] M. F. Schwartz, M. W. Montgomery, L. J. Buxbaum, S. S. Lee, T. G. Carew, H. Coslett, M. Ferraro, E. Fitzpatrick-DeSalme, T. Hart, and N. Mayer. Naturalistic action impairment in closed head injury. *Neuropsychology*, 12(1):13, 1998.
- [12] M. B. Holm and J. C. Rogers. The performance assessment of self-care skills. In *Assessments in occupational therapy mental health*, pages 359–370. Routledge, 2024.
- [13] D. A. Norman and T. Shallice. *Attention to Action: Willed and Automatic Control of Behavior*, pages 1–18. Springer US, 1986. ISBN 9781475706291. doi:10.1007/978-1-4757-0629-1_1. URL http://dx.doi.org/10.1007/978-1-4757-0629-1_1.
- [14] R. Cooper and T. Shallice. Contention scheduling and the control of routine activities. *Cognitive neuropsychology*, 17(4):297–338, 2000.
- [15] R. P. Cooper and T. Shallice. Hierarchical schemas and goals in the control of sequential behavior. *Psychological Review*, 113(4):887–916, 2006. ISSN 0033-295X. doi:10.1037/0033-295X.113.4.887. URL <http://dx.doi.org/10.1037/0033-295X.113.4.887>.
- [16] M. F. Schwartz. The cognitive neuropsychology of everyday action and planning. *Cognitive neuropsychology*, 23(1):202–221, 2006.

- [17] M. F. Schwartz, E. S. Reed, M. Montgomery, C. Palmer, and N. H. Mayer. The quantitative description of action disorganisation after brain damage: A case study. *Cognitive Neuropsychology*, 8(5):381–414, 1991.
- [18] M.-A. Côté, A. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, M. Hausknecht, L. El Asri, M. Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018.
- [19] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.
- [20] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502, 2018.
- [21] X. Puig, T. Shu, S. Li, Z. Wang, Y.-H. Liao, J. B. Tenenbaum, S. Fidler, and A. Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration. *arXiv preprint arXiv:2010.09890*, 2020.
- [22] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on robot learning*, pages 477–490. PMLR, 2022.
- [23] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.
- [24] Y. Wan, J. Mao, and J. Tenenbaum. Handmethat: Human-robot communication in physical and social environments. *Advances in Neural Information Processing Systems*, 35:12014–12026, 2022.
- [25] D. Ghose, O. Gitelson, M. Vazquez, and B. Scassellati. Open-ended goal inference through actions and language for human-robot collaboration. In *Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction*, pages 68–77, 2026.
- [26] M. Botvinick and D. C. Plaut. Doing without schema hierarchies: A recurrent connectionist approach to normal and impaired routine sequential action. *Psychological Review*, 111(2): 395–429, 2004. ISSN 0033-295X. doi:10.1037/0033-295x.111.2.395. URL <http://dx.doi.org/10.1037/0033-295X.111.2.395>.
- [27] S. C. Seligman, T. Giovannetti, J. Sestito, and D. J. Libon. A new approach to the characterization of subtle errors in everyday action: Implications for mild cognitive impairment. *The Clinical Neuropsychologist*, 28(1):97–115, Nov. 2013. ISSN 1744-4144. doi:10.1080/13854046.2013.852624. URL <http://dx.doi.org/10.1080/13854046.2013.852624>.
- [28] A. Mihailidis, J. N. Boger, T. Craig, and J. Hoey. The coach prompting system to assist older adults with dementia through handwashing: An efficacy study. *BMC Geriatrics*, 8(1), Nov. 2008. ISSN 1471-2318. doi:10.1186/1471-2318-8-28. URL <http://dx.doi.org/10.1186/1471-2318-8-28>.
- [29] J. Hoey, P. Poupart, A. v. Bertoldi, T. Craig, C. Boutilier, and A. Mihailidis. Automated hand-washing assistance for persons with dementia using video and a partially observable markov decision process. *Computer Vision and Image Understanding*, 114(5):503–519, May 2010. ISSN 1077-3142. doi:10.1016/j.cviu.2009.06.008. URL <http://dx.doi.org/10.1016/j.cviu.2009.06.008>.

- [30] K. Hackett, M. McKniff, S. Lehman, E. Pinsky, C. C. Tan, M. Kaplan, G. Vallecorsa, M. D. Serruya, and T. Giovannetti. A counterbalanced crossover pilot study of a personalized smart-phone reminder application for older adults with cognitive impairment. *Neuropsychological Rehabilitation*, 36(1):1–35, Apr. 2025. ISSN 1464-0694. doi:10.1080/09602011.2025.2489125. URL <http://dx.doi.org/10.1080/09602011.2025.2489125>.
- [31] R. Peddi, S. Arya, B. Challa, L. Pallapothula, A. Vyas, B. Gouripeddi, Q. Zhang, J. Wang, V. Komaragiri, E. Ragan, et al. Captaincook4d: A dataset for understanding errors in procedural activities. *Advances in Neural Information Processing Systems*, 37:135626–135679, 2024.
- [32] K. Yordanova, A. Hein, and T. Kirste. Kitchen task assessment dataset for measuring errors due to cognitive impairments. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2020.
- [33] A. Flaborea, G. Di Mattia, A. Furnari, G. M. Farinella, and F. Galasso. PREGO: Online mistake detection in PProcedural EGocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18483–18492, 2024.
- [34] L. Seminara, G. M. Farinella, and A. Furnari. Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos. In *Advances in Neural Information Processing Systems*, volume 37, 2024.
- [35] S. Majumder, A. Nethi, Z. Al-Halah, and K. Grauman. MistExit: Learning to exit for early mistake detection in procedural videos. *arXiv preprint arXiv:2603.14252*, 2026.
- [36] M. Ramírez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1121–1126, 2010.
- [37] C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, Dec. 2009. ISSN 0010-0277. doi:10.1016/j.cognition.2009.07.005. URL <http://dx.doi.org/10.1016/j.cognition.2009.07.005>.
- [38] T. Zhi-Xuan, J. L. Mann, T. Silver, J. B. Tenenbaum, and V. K. Mansinghka. Online Bayesian goal inference for boundedly rational planning agents. In *Advances in Neural Information Processing Systems*, volume 33, pages 19238–19250, 2020.
- [39] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2022.
- [40] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, pages 1769–1782. PMLR, 2022.
- [41] Y. Wan, Y. Wu, Y. Wang, J. Mao, and N. Jaques. Infer human’s intentions before following natural language instructions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25309–25317, 2025.
- [42] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.
- [43] M. P. Lawton and E. M. Brody. Assessment of older people: self-maintaining and instrumental activities of daily living. *The gerontologist*, 1969.

- [44] C. M. Baum, L. T. Connor, T. Morrison, M. Hahn, A. W. Dromerick, and D. F. Edwards. Reliability, validity, and clinical utility of the executive function performance test: A measure of executive function in a sample of people with stroke. *The American journal of occupational therapy*, 62(4):446–455, 2008.
- [45] S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson, and M. W. Jaffe. Studies of illness in the aged: the index of adl: a standardized measure of biological and psychosocial function. *jama*, 185(12):914–919, 1963.

Neuropsychological grounding of the error taxonomy

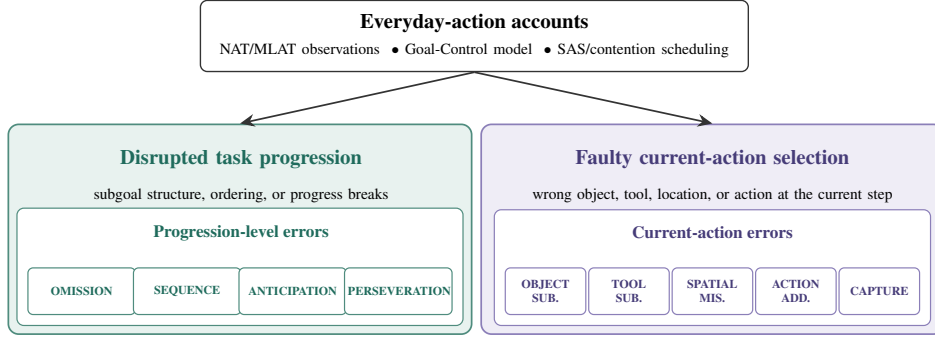


Figure 4: The taxonomy is organized by what an error disrupts: progression-level errors break the subgoal structure and ordering of the task, whereas current-action errors corrupt the selection of the current object, tool, location, or action. This organizing axis is distinct from the Goal-Control generator’s factorization (goal activation vs. executive control; Section A.6), which we use to sample errors.

This appendix provides the additional details for the main paper. Section A describes how the benchmark is built, from its neuropsychological grounding to the symbolic domain and the Goal-Control error generator. Section B reports dataset statistics and the data split. Section C covers the baseline and human-study implementations together with the complete LLM results.

A Benchmark Design

A.1 Neuropsychology of Everyday Action

HELPMEOU builds its error labels and generation model on prior work in clinical neuropsychology on naturalistic action. We draw on three lines of work.

Performance-based instruments. The Naturalistic Action Test (NAT) and the Multi-Level Action Test (MLAT) ask participants to perform familiar multi-object tasks such as making toast and coffee, wrapping a gift, packing a lunchbox—and score the resulting errors against a standardized taxonomy of omissions, sequence errors, substitutions, additions, and perseverations [10, 11]. These instruments establish that everyday-action failures are structured and recurrent across clinical populations (left-CVA, right-CVA, traumatic brain injury, dementia); this structure makes everyday-action errors suitable for systematic monitoring.

The Supervisory Attentional System (SAS) and contention scheduling. Contention-scheduling accounts describe routine action as competition among activated schemas, while the Supervisory Attentional System (SAS) provides top-down control when routine schema selection is insufficient [13, 14, 15]. Within this view, weak or decaying task goals can manifest as omissions, whereas poor suppression of competing schemas can manifest as commission errors, especially around subgoal transitions [16, 17].

The Goal-Control account. Giovannetti et al. [9] synthesize these observations into a Goal-Control model that separates failures of goal activation from failures of executive control. We use this two-factor account in the generator (Section A.6), but the error taxonomy itself is organized by what an error disrupts: the progression of the task or the selection of the current action. Figure 4 summarizes this organizing axis, and the per-category clinical error rates we calibrate against are reported in Table 4.

Table 4: Per-category error proportions used by the HELPMEOU generator in the nine-category taxonomy: (a) the omission–commission split and (b) the commission-category weights w_c . We set these proportions from everyday-action error rates reported for clinical populations [10, 9, 2], pooling the left-CVA ($n=16$), right-CVA ($n=30$), TBI ($n=30$), and dementia ($n=54$) groups. We then make small simulator-specific adjustments and add categories supported by the broader action-error literature [13]. Because an error type can be realized only when it is feasible (Section A.6), the released episodes were selected per category so that the released distribution approximates these targets (Figure 7).

<i>(a) Omission vs. commission (share of all errors)</i>	
Omission	0.44
Commission	0.56
<i>(b) Commission category weight w_c (sums to 1.00)</i>	
Sequence	0.22
Object substitution	0.17
Perseveration	0.15
Capture	0.13
Spatial misreach	0.13
Anticipation	0.10
Tool substitution	0.05
Action addition	0.05

Table 5: Object hierarchy. Five classes spanning 187 types; the 156 movable types (RECEPTACLE/FOOD/TOOL/THING) are the “object types” of the main text, and the 31 LOCATION types are the non-movable ontop/inside locations.

Class	#Types	Subclasses
LOCATION	31	ontop, inside (floor, countertop, table, desk, shelf, stove, bed, sofa, toilet, cabinet, refrigerator, microwave, oven, dishwasher, sink, washing_machine, dryer, toaster, closet, drawer, medicine_cabinet, bathtub, ...)
RECEPTACLE	31	tableware, vessel, cooking_vessel, bag, box, basket, tray, bucket, ashcan
FOOD	49	fruit, vegetable, drink, protein, bread_grain, spread, condiment, sauce, prepared_food, snack
TOOL	31	cutlery, cleaning_tool, cleansing, wrapping_tool, writing_tool, toiletry, laundry_tool
THING	45	clothing, linen, soft_furnishing, footwear, headwear, accessory, wrapping_material, paper_product, personal_item, medication

A.2 Object Hierarchy and Symbolic Domain

HELMEOU represents each household scene with an object-centric symbolic state. All entities are organized into a five-class hierarchy (Table 5). The LOCATION class contains the 31 non-movable entities an agent can move to, grouped by whether objects can be placed ontop of them or inside them; the remaining four classes contain the 156 movable object types referred to in the main text. Including locations, the domain defines 187 object types in total; the 156 “object types” quoted in the main text counts only movable objects (RECEPTACLE, FOOD, TOOL, THING).

Predicates. The state is described with spatial relations (inside, ontop, human-at, holding, hand-empty), boolean state attributes (open, toggled, cooked, frozen, sliced, soaked, dusty, stained, toasted, buttered, stirred, folded, ironed, hung, worn, wrapped,

packed, taken, ...), affordance meta-predicates (movable, openable, toggleable, cookable, freezable, sliceable, soakable, foldable, hangable, wearable, toastable, ...), and per-type predicates (type-knife, type-spoon, type-iron, ...) used to express tool/object identity constraints. Spatial relations are represented directly (only the immediate container/surface), so moving a container automatically carries its contents.

A.3 Action Operators

The domain defines 38 action schemas (Table 6), each specified in PDDL with full preconditions and effects; we give intuitive semantics here and release the complete domain file.

Table 6: The 38 action operators in HELPMEOU. The semantics column gives an informal description of each operator’s effect.

Operator (& key arguments)	Semantics
human-move (from,to)	Move between two non-movable locations.
human-pick-up-at-location (obj,loc)	Pick an object off a location (hand must be empty).
human-pick-up-from-receptacle-at-location (obj,rec,loc)	Pick an object out of/off a receptacle at a location.
human-put-inside-location (obj,loc)	Put the held object inside a location.
human-put-ontop-location (obj,loc)	Put the held object on top of a location.
human-put-inside-receptacle-at-location (obj,rec,loc)	Put the held object inside a receptacle.
human-put-ontop-receptacle-at-location (obj,rec,loc)	Put the held object on top of a receptacle.
human-open-location (loc)	Open an openable location.
human-close-location (loc)	Close an open location.
human-open-receptacle-at-location (rec,loc)	Open an openable receptacle.
human-close-receptacle-at-location (rec,loc)	Close an open receptacle.
human-toggle-on-location (loc)	Toggle a toggleable appliance on.
human-toggle-off-location (loc)	Toggle a toggleable appliance off.
human-heat-obj (obj,loc)	Heat/cook an object at a microwave/oven.
human-heat-obj-on-stove (obj,vessel,loc)	Cook an object in a vessel on a stove.
human-cool-obj (obj,loc)	Freeze an object inside a refrigerator.
human-slice-obj (obj,tool,loc)	Slice a sliceable object with a knife.
human-soak-obj (obj,loc)	Soak a soakable object at a sink.
human-clean-obj-at-location (obj,tool,loc)	Remove dust/stain from an object with a cleaning tool.
human-clean-location (tool,loc)	Remove dust/stain from a location.
human-discard-obj (obj,bin,loc)	Throw an object into a bin.
human-replace-obj (new,old,loc)	Swap a new object in for an old one on a surface.

Table 6 (continued)	
Operator (& key arguments)	Semantics
human-toast-obj (obj, loc)	Toast a toastable object (sets toasted, cooked).
human-spread (obj, substance, tool, loc)	Spread a substance (butter/jam) onto an object with a tool.
human-pour (content, source, dest, loc)	Pour content from a source vessel into a destination.
human-stir (container, tool, loc)	Stir a container with a spoon.
human-wrap-obj (obj, material, tool, loc)	Wrap an object in wrapping material.
human-close-gift-box-flaps (box, loc)	Close the flaps of a gift box.
human-close-gift-box-lid (box, loc)	Close the lid of a gift box.
human-pack-obj (obj, container, loc)	Pack an object into a container.
human-wear-obj (obj)	Put on a wearable object.
human-remove-worn-obj (obj)	Take off a worn object.
human-fold-obj (obj, loc)	Fold a foldable object.
human-iron-obj (obj, tool, loc)	Iron an object with an iron.
human-hang-obj (obj, loc)	Hang a hangable object.
human-wash-clothes (obj, loc)	Wash clothing at a washing machine.
human-dry-clothes (obj, loc)	Dry clothing at a dryer.
human-take-medication (obj)	Take a medication object.

Action constraints. Several operators carry tool/location constraints enforced through the per-type predicates: slicing requires a `type-knife` tool, stirring a `type-spoon` tool, ironing a `type-iron` tool; heating is only valid at a microwave/oven/stove, cooling only inside a refrigerator, soaking only at a sink, and cleaning only with a suitable cleaning tool. These constraints are exactly the preconditions whose violation defines the `TOOL-SUBSTITUTION` error class (Section A.5).

A.4 Task Templates

HELPMEOU spans 50 ADL/IADL task templates organized into eight families (Table 7). Each template is anchored in a clinical assessment, functional scale, or occupational-therapy framework for everyday action—NAT, PASS, the Lawton IADL scale, the Katz ADL index, the Executive Function Performance Test (EFPT), or the OTPF-4 occupational-therapy framework—and is annotated with a complexity tier (simple / medium / complex) reflecting the number of accomplishment steps. Each template is expressed as a set of first-order subgoals over the predicates of Section A.2; Figure 5 shows a worked example.

Table 7: The 50 ADL/IADL task templates, grouped into eight families.

#	Template	Anchor	Complexity
<i>F1: Food preparation</i>			
0	making_toast_and_coffee	NAT	complex
1	slicing_fruit_and_refrigerating	NAT	medium
2	cooking_and_plating_protein	NAT	simple
3	storing_fruits_and_vegetables	Lawton IADL	simple
4	making_a_salad	NAT	medium
5	preparing_breakfast	NAT	complex
<i>F2: Gift wrapping</i>			

Table 7 (continued)

#	Template	Anchor	Complexity
6	wrapping_a_book	PASS	simple
7	wrapping_two_items	PASS	simple
8	wrapping_book_and_boxing	PASS	medium
9	wrapping_shirt_and_bagging	PASS	medium
<i>F3: Packing</i>			
10	packing_school_bag_and_lunch	Lawton IADL	complex
<i>F4: Table setting & serving</i>			
11	setting_table_and_serving_meal	PASS	complex
12	setting_a_simple_table	PASS	simple
13	plating_a_cooked_dish	OTPF-4	simple
14	preparing_a_tea_tray	Lawton IADL	medium
15	setting_table_for_two	PASS	medium
16	serving_breakfast_on_tray	OTPF-4	medium
17	clearing_the_dining_table	Lawton IADL	medium
<i>F5: Cleaning & tidying</i>			
18	cleaning_kitchen_and_storing_dishes	PASS	complex
19	cleaning_bathroom	OTPF-4	medium
20	cleaning_dining_room	OTPF-4	medium
21	wiping_a_countertop	PASS	simple
22	washing_dishes_and_putting_away	Lawton IADL	simple
23	doing_dishes_and_cleaning_sink	PASS	medium
24	tidying_a_desk	OTPF-4	medium
25	organizing_a_bookshelf	OTPF-4	medium
26	cleaning_living_room	Lawton IADL	medium
27	tidying_a_bedroom	OTPF-4	complex
<i>F6: Laundry & clothing care</i>			
28	folding_and_storing_clothes	Lawton IADL	medium
29	hanging_clothes_and_folding_linens	Lawton IADL	medium
30	ironing_folding_storing_shirt	Lawton IADL	medium
31	folding_and_storing_linens	Lawton IADL	medium
32	folding_clothes_and_ironing	Lawton IADL	medium
33	putting_away_folded_clothes	Lawton IADL	simple
34	hanging_a_shirt_to_dry	Lawton IADL	simple
35	sorting_laundry_into_baskets	Lawton IADL	medium
<i>F7: Medication & personal care</i>			
36	preparing_and_taking_medication	EFPT	medium
37	taking_a_pill	EFPT	simple
38	filling_a_glass_of_water	PASS	simple
39	organizing_weekly_pill_box	EFPT	medium
40	setting_up_toiletries	PASS	medium
41	restocking_bathroom_supplies	OTPF-4	complex
<i>F8: Daily routines</i>			
42	preparing_to_leave_home	OTPF-4	medium
43	putting_on_shoes_and_grabbing_keys	Katz ADL	simple
44	getting_dressed	Katz ADL	simple
45	dressng_for_rain	OTPF-4	medium
46	preparing_for_work	Lawton IADL	complex
47	packing_a_gym_bag	OTPF-4	medium
48	coming_home_routine	OTPF-4	medium
49	preparing_for_bed	OTPF-4	medium

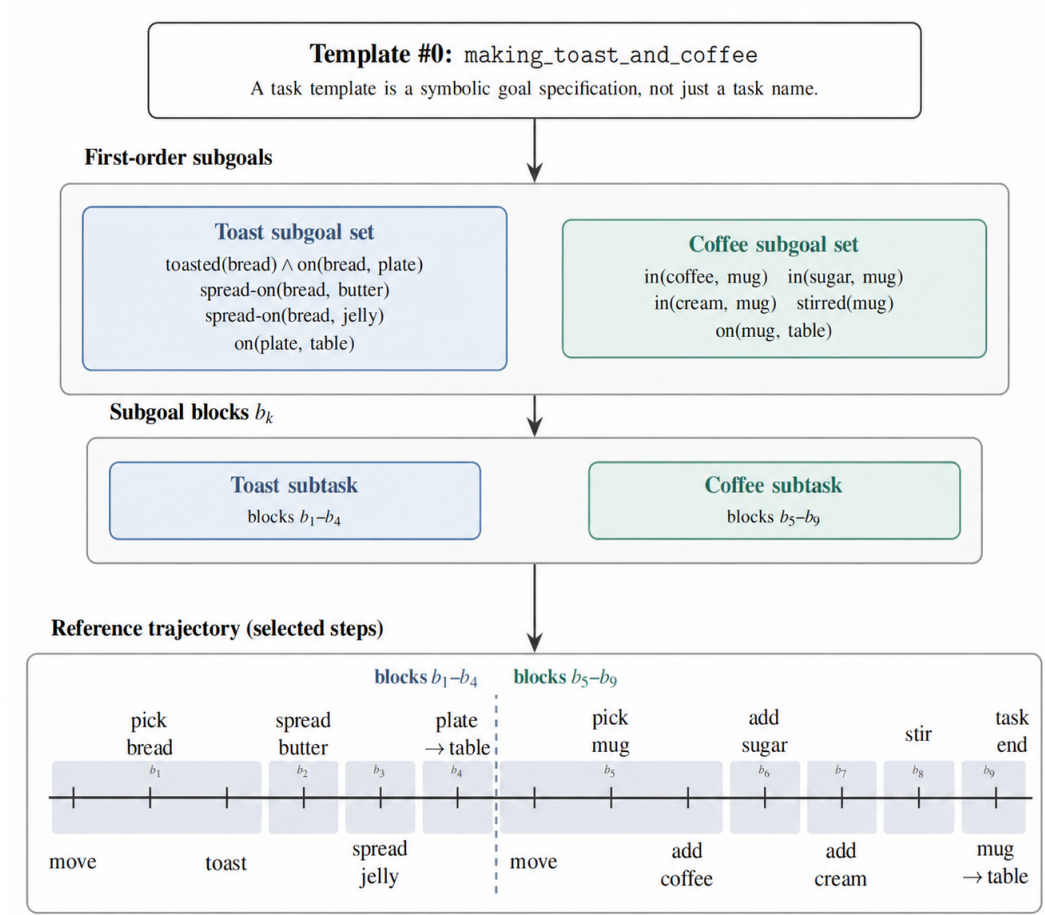


Figure 5: An example of one executable task template. Template #0 (making_toast_and_coffee) expands into first-order subgoals—the subgoal blocks b_k , here grouped into a toast and a coffee subtask—which a planner solves into a correct reference trajectory (the shaded bands are the b_k). The block boundary shown on the trajectory is the structure the generator uses for progression-level errors; in generated episodes the block order is randomized, and onsets are labelled by the category rules of Section A.5. Only representative actions are shown.

A.5 Error Taxonomy

The nine error categories are organized into two levels. Progression-level errors disrupt the structure of task execution; current-action errors affect local action selection, object use, or spatial targeting. For each category we give its definition, trigger condition, and where the onset is labelled; Figure 6 pairs a correct and an errorful micro-example per category.

Each definition below names the step at which the onset is labelled; onsets are recognizable from the observed prefix and the task, without the reference execution. An immediately repeated action without additional effect, or walking back and forth between places the task uses, is harmless redundancy and not an error.

Progression-level errors.

- **OMISSION.** An entire subgoal block is deleted. Because correct executions may perform subgoals in different orders, the missing subgoal could still be performed later; the omission is therefore confirmed at the task-end event. *Onset:* the task-end step. When the missing step has an earlier observable consequence, such as storing an item before it is processed,

Error categories: correct vs. errorful micro-examples

Progression-level errors <i>errors that disrupt task structure or ordering</i>	Current-action errors <i>errors in selecting the current object/tool/location/action</i>
1. Omission <i>a required step is never done (confirmed at task end).</i> Correct: add coffee → stir → serve → task end Error: add coffee → serve → task end ← onset	5. Object substitution <i>a similar object of the same kind is used.</i> Correct: spread butter on bread Error: spread jam on bread
2. Sequence <i>two steps that must occur in order are swapped.</i> Correct: add coffee → add sugar → stir Error: add coffee → stir → add sugar	6. Tool substitution <i>a similar tool is used instead of the right one.</i> Correct: stir with spoon Error: stir with fork
3. Anticipation <i>an item is stored before it is processed.</i> Correct: fold shirt → put shirt in closet Error: put unfolded shirt in closet (never folded)	7. Spatial misreach <i>opens an unrelated storage place and leaves empty-handed.</i> Correct: go to cabinet → take mug Error: go to drawer → open drawer → leave empty-handed
4. Perseveration <i>an action is repeated after doing something else.</i> Correct: add sugar → butter toast → serve Error: add sugar → butter toast → add sugar again	8. Action addition <i>an unrelated item is put into the task's container.</i> Correct: put sandwich in lunchbox Error: put sandwich in lunchbox → put sock in lunchbox
	9. Capture <i>a needed action is applied to an unrelated object.</i> Correct: wrap book → put book in gift box Error: wrap phone → (book left unwrapped)

Figure 6: A correct-vs-errorful micro-example for each of the nine error categories, grouped into progression-level and current-action errors. In every ERROR row, the step(s) that diverge from the correct execution are highlighted in red. Examples use the household domain of Figure 5.

the error is labelled ANTICIPATION and the deleted step is not counted as an omission, following MLAT coding [11].

- **SEQUENCE.** Two steps that must occur in a specific order are performed in the wrong order. Two constructions are used: stirring before the last ingredient is added (the MLAT example “stir mug of water, then add grinds”), and violating a step-order norm (buttering bread before toasting it, folding before ironing, taking a pill before pouring water, putting on a jacket before the required shirt). Other orders that reach the goal are correct. *Onset:* the step performed out of order.
- **ANTICIPATION.** An item is stored, served, or packed before it receives the processing the task requires, or a gift box’s lid is closed without first closing its flaps, which are never closed afterwards. *Onset:* the step that commits the unprocessed item: putting it into its final place, closing the container that holds it, or delivering that container to its final place.
- **PERSEVERATION.** After doing something else in between, the person repeats an action that should be done only once, such as adding sugar to the coffee again after buttering the toast, or taking an extra dose of a pill the task does not require. *Onset:* the repeated action.

Current-action errors.

- **OBJECT_SUBSTITUTION.** A confusable object of the same subclass is used in place of the object the task needs (e.g. jam for butter). *Onset:* the first step that uses the substitute.
- **TOOL_SUBSTITUTION.** A confusable tool is used in place of the appropriate one (e.g. a fork where a spoon is required). *Onset:* the first step that uses the substitute tool.
- **ACTION_ADDITION.** An item unrelated to the task (not named in the goal and of a different subclass from every task object) is put into or onto the task’s target container, plate, or tray. Removing it later does not undo the error. *Onset:* the step that puts it there.

Table 8: Generator hyperparameters.

Symbol / name	Value	Meaning
θ_{goal}	0.48	baseline goal salience
θ_{decay}	0.20	per-subgoal activation decay (cognitive load)
θ_{ctrl}	0.12	commission-error probability
τ_{omit} (OMIT_THRESHOLD)	0.30	omission activation threshold
σ (ACTIVATION_NOISE_STD)	0.10	activation noise std. dev.
min. trajectory length	10	shorter trajectories are filtered

- **CAPTURE.** An action the task needs is applied to an unrelated object of a different subclass instead of the task’s object, which is then left unprocessed. *Onset:* the action applied to the unrelated object (fetching it or walking to it does not count).
- **SPATIAL_MISREACH.** The person goes to a storage location that the task never uses, opens it, and leaves without taking anything. *Onset:* the step of leaving that location empty-handed, the first step at which the visit is recognizably unrelated to the task.

A.6 Error-Generation Model

Goal-Control process (supplementary parameter values). The main text defines the Goal-Control process (Eq. 2): for each subgoal block b_k an activation $u_k = \theta_{\text{goal}}(1 - \theta_{\text{decay}})^k + \epsilon_k$ gates omission when $u_k < \tau_{\text{omit}}$, and surviving blocks receive a commission with probability θ_{ctrl} . Here we report only the values deferred to this supplement: the omission threshold $\tau_{\text{omit}}=0.3$ and the activation noise $\sigma=0.1$ (with $\epsilon_k \sim \mathcal{N}(0, \sigma^2)$). The impaired profile $(\theta_{\text{goal}}, \theta_{\text{decay}}, \theta_{\text{ctrl}})=(0.48, 0.20, 0.12)$ and the omission-vs-commission split it targets are reported in Table 4; all generator hyperparameters are collected in Table 8.

Two-axis per-subgoal decision. The generator walks the subgoal blocks of the reference trajectory in order and applies two decisions at each block. (i) *Goal axis:* it first tests the activation u_k ; if $u_k < \tau_{\text{omit}}$ the block is dropped as an OMISSION. (ii) *Control axis:* otherwise, with probability θ_{ctrl} a commission is injected, and its category is drawn from the weights w_c (Table 4(b)) restricted to the categories that are feasible at the current step. These two axes correspond to the two panels of Table 4: the goal-vs-control decision gives the omission-vs-commission split (a), and the weighted draw on the control axis gives the commission-category weights (b). Because a commission category is realized only when feasible in the current state, the dataset frequencies can differ from w_c (see *Per-category feasibility* below).

Per-category feasibility $\phi_c(s_k)$. Not every commission category can be injected at every step. Each injector requires a particular trajectory or scene configuration: for example, OBJECT_SUBSTITUTION requires a confusable alternative object in the scene, while SEQUENCE requires reorderable subgoal blocks. We represent this with a binary feasibility gate $\phi_c(s_k)$, which is 1 only when category c can be realized at step k . As a result, the observed category frequencies are shaped by both the sampling weights w_c and the states in which each error is feasible. Table 9 summarizes these gates. Every injected episode must also pass the validation of Section A.7, match its category rule exactly once at the injected step, and the reference execution must not match the rule.

Calibration. We calibrate $\theta_{\text{goal}}, \theta_{\text{decay}}, \theta_{\text{ctrl}}$ and w_c so that the marginal error distribution produced by the generator matches the per-category rates of Table 4. Figure 7 compares the generated distribution against the clinical targets. Because the feasibility gates bind unevenly, the raw generator output under-represents low-feasibility categories; the released episodes were therefore selected per category from a larger generated pool, so that the released distribution approximates the targets.

Table 9: Per-category feasibility gate $\phi_c(s_k)$: the structural condition under which each error type can be injected at step k . A sampled error is realized only when its condition holds in the current state; otherwise the generator resamples.

Category	Feasible when ...
OMISSION	a subgoal block can be deleted while the trajectory stays executable
SEQUENCE	a stir follows several additions to one container, or a step-order norm applies
ANTICIPATION	the task requires processing an item before it is stored, served, or packed, or requires closing a gift box’s flaps before its lid
PERSEVERATION	an addition (or dose) can be repeated after intervening actions, with all goals still met
OBJECT_SUBSTITUTION	a same-subclass confusable object is present in the scene
TOOL_SUBSTITUTION	a confusable alternative tool is present in the scene
SPATIAL_MISREACH	an openable storage location exists that the task never uses
ACTION_ADDITION	an item unrelated to the task can be put into the task’s target container
CAPTURE_ERROR	an unrelated object of a different subclass affords a processing action the task needs

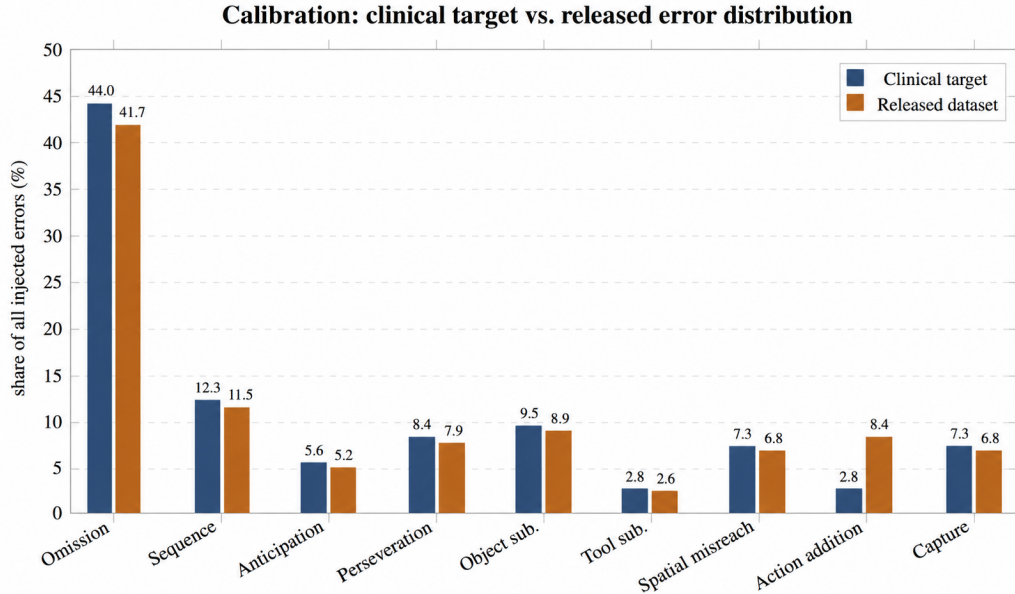


Figure 7: Calibration check: the generator targets the clinical per-category rates of Table 4 (blue); the released distribution (orange) is within 2.3 percentage points of the target for every category except action addition (8.4% vs. 2.8%), which frequently occurs as the second error of a multi-error episode. Shares are over all injected errors (12,817 in the release) and each series sums to 100%.

A.7 Dual-Standard Validation

A core requirement is that errorful trajectories be *semantically wrong yet physically executable*. We therefore validate with two standards.

Strict standard (7 categories). OMISSION, SEQUENCE, ANTICIPATION, PERSEVERATION, ACTION_ADDITION, CAPTURE_ERROR, and SPATIAL_MISREACH must pass a complete PDDL replay: every action’s preconditions must hold in the simulated state. These errors are “legal-but-wrong” executions and remain fully physically consistent.

Dataset statistics: released HELPMEOU benchmark

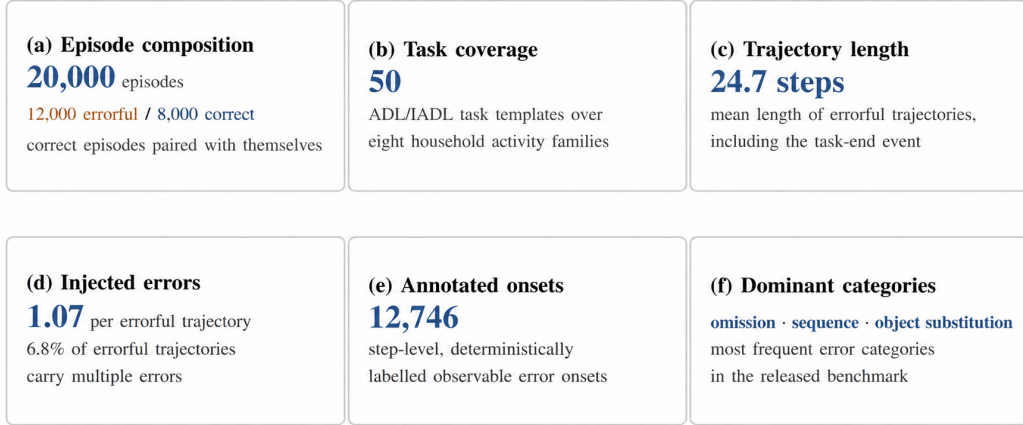


Figure 8: Summary statistics of the released HELPMEOU benchmark: (a) episode composition (paired errorful/correct), (b) task coverage, (c) mean trajectory length, (d) injected errors per trajectory, (e) annotated step-level onsets, and (f) the most frequent error categories.

Relaxed standard (2 categories). TOOL_SUBSTITUTION and OBJECT_SUBSTITUTION would be rejected by a naive validator—using the wrong tool/object is the error. For these we enforce only the physical chain—navigation, accessibility, containment, and hand occupancy—while relaxing the single semantic precondition that defines the substitution (e.g. the type-spoon requirement of human-stir). Within the error block, only the semantic type constraint is relaxed; all physical preconditions still hold.

Confusable-alternate selection. Substituted tools/objects are drawn from a hierarchy-constrained confusable set: TOOL/FOOD/THING substitutions stay within the same subclass, RECEPTACLE substitutions may cross subclass, and no substitution ever crosses a parent class. This keeps each substitution interpretable as a human confusion rather than an arbitrary swap.

B Dataset Statistics

B.1 Corpus Overview

Figure 8 summarizes the released HELPMEOU corpus—its episode composition, task coverage, trajectory length, injected-error rate, annotated onsets, and dominant error categories.

B.2 Data Split

The released corpus contains 20,000 episodes over 50 templates: 17,478 for training, 2,022 for validation, and 500 for testing. The errorful test episodes are stratified by category in proportion to the full corpus and the test set covers all 50 templates (312 errorful, 188 correct; 121 simple, 291 medium, and 88 complex episodes by template tier). All learned and non-learned baselines are evaluated on the full test set. Because frontier-LLM evaluation is substantially more expensive, LLMs and the human reference are evaluated on a fixed 60-episode subset of the test set with 30 errorful episodes, covering every category (three to four per category), and 30 correct episodes from 30 different templates; the learned and non-learned baselines are re-scored on the same subset for comparison.

Table 10: Evaluation splits used in the main paper. The 60-episode LLM subset is drawn from the held-out test set; all other baselines are reported both on the full test set and on this common subset for fair comparison with LLMs.

Split	Episodes	Use
Released corpus	20,000	Full benchmark release
Training / validation	17,478 / 2,022	Training and checkpoint selection
Held-out test set	500	Main baseline evaluation
LLM subset	60	Frontier-LLM and human-reference evaluation and matched baseline subset scores

C Experimental Details

C.1 Learned Baselines

The learned baselines are trained as discriminative error monitors on the benchmark’s training corpus (disjoint from the held-out test set) and run as online monitors on the 500-episode test set. Every learned baseline receives the same input as the LLMs: the task name, the initial scene, and the action prefix; none receives the formal goal, the reference execution, or any label.

Seq2Seq. At each step, the text-only baseline encodes the task description, the initial scene description, and the observed action prefix as token sequences. The shared sequence representation feeds three supervised prediction heads: a step-level error head producing p_t (the reported p_instant score), a 9-class multi-label error-type head producing p_type, and an auxiliary prefix-error head used only as training supervision. The auxiliary prefix head is not used by the reported metrics.

Training minimizes the sum of binary cross-entropy for the step-level error head, multi-label binary cross-entropy for the type head, and auxiliary binary cross-entropy for the prefix head. The step-level and prefix heads are trained without positive-class weights; the type head uses per-class positive weights equal to the negative-to-positive ratio. We train with batch size 32 (two accumulated batches of 16), learning rate 10^{-4} , and a single seed (42), evaluate on the validation split every 2,000 steps, and keep the checkpoint with the highest validation M2 AUPRC; the test set is scored once.

The reported metrics are derived directly from these outputs: M1 uses $\max_t p_t$, M2 is the AUPRC of p_t against the step label y_t , M3 is the first time step with $p_t \geq \theta$, and M4 uses the 9-class p_type predictions.

Seq2Seq+SG. The +SG variant augments the same Seq2Seq backbone with an explicit scene-graph encoder. A relational graph is built from the symbolic scene state after replaying the observed prefix in the PDDL simulator (object identity, location, state, and containment; there is no geometry, hence the renaming from “+3D”) (`build_local_graph`); each node carries a 134-dimensional feature (a 32-d learned type embedding, plus class, subclass, affordance, and state-boolean one-hot blocks), and a two-layer R-GCN ($\text{RGCNConv}(134, 64) \rightarrow \text{ReLU} \rightarrow \text{Dropout}(0.2) \rightarrow \text{RGCNConv}(64, 64) \rightarrow \text{ReLU}$) with four relation types and mean-pool readout produces a graph-level embedding that is fused before the prediction heads. Training otherwise matches the Seq2Seq baseline (same discriminative multi-head loss). This isolates the contribution of explicit object/spatial state to error monitoring. Both supervised baselines serve as reference implementations rather than new model contributions. Seq2Seq+SG performs slightly better than Seq2Seq (main paper, Table 2).

Per-class error-type diagnosis (M4). Table 11 reports per-class M4 F1 of the learned monitors on the 500-episode test set. M4 scores the cumulative category predictions in errorfull episodes from the first onset onward and in correct episodes at every evaluated step, where naming any category is a false positive; each episode is weighted equally. The type heads of the learned monitors name an

Table 11: Per-class M4 F1 ($\theta=0.5$) of the learned monitors on the 500-episode test set, with correct episodes included (true category set empty). The last row gives macro-F1 when only errorful episodes are scored.

Error category	Seq2Seq	Seq2Seq+SG	SGT
Omission	0.893	0.888	0.916
Sequence	0.778	0.741	0.921
Anticipation	0.386	0.422	0.607
Perseveration	0.632	0.664	0.803
Object substitution	0.145	0.166	0.196
Tool substitution	0.113	0.122	0.000
Action addition	0.411	0.691	0.216
Capture	0.708	0.982	0.502
Spatial misreach	0.456	0.254	0.715
macro-F1	0.502	0.548	0.542
macro-F1, errorful episodes only	0.631	0.678	0.638

Table 12: Per-class step-level AUPRC (M2) of the learned monitors on the 500-episode test set. Each row pools the test episodes whose primary category is the given one with the 188 correct episodes; n is the number of episodes and of error steps (multi-error episodes contribute all their error steps). Random equals the positive-step rate of the row.

Error category	n (episodes / steps)	Random	Seq2Seq	Seq2Seq+SG	SGT
Omission	137 / 137	0.026	0.696	0.716	0.730
Sequence	38 / 38	0.010	0.186	0.215	0.814
Anticipation	17 / 17	0.005	0.023	0.085	0.359
Perseveration	26 / 26	0.007	0.093	0.095	0.502
Object substitution	30 / 30	0.008	0.018	0.012	0.038
Tool substitution	9 / 9	0.003	0.003	0.003	0.020
Action addition	9 / 9	0.003	0.005	0.024	0.029
Capture	23 / 44	0.012	0.204	0.189	0.178
Spatial misreach	23 / 23	0.007	0.108	0.052	0.771

error category in most correct episodes (at the last step: all 188 for Seq2Seq and Seq2Seq+SG, 160 for SGT), mostly omission and object substitution, which lowers their macro-F1 from 0.63–0.68 when scored on errorful episodes alone to 0.50–0.55. Omission and sequence remain the best-diagnosed categories ($F1 \geq 0.74$). SGT is best on anticipation, perseveration, and spatial misreach, the Seq2Seq models on capture and action addition; object and tool substitution stay at or below 0.20 for all three models. Predicting categories from the task name alone, using their training frequencies per template, reaches at most 0.20 in the errorful-only scoring, so the type heads use the observed actions rather than a task prior.

Per-class step-level detection (M2). Table 12 breaks the step-level AUPRC of the learned monitors down by error category. For each category we pool the evaluation-window steps of the test episodes of that category with those of all 188 correct episodes; for omissions all evaluated steps are scored, with the task-end step as the positive, and for all other categories the steps before task end. SGT’s advantage concentrates on categories with a local action signature—SEQUENCE (0.814), SPATIAL_MISREACH (0.771), and PERSEVERATION (0.502)—which it can recognize from the current action and the scene graph that precedes it. Categories whose detection requires knowing which object, tool, or item the task calls for remain near chance for all three models: OBJECT_SUBSTITUTION (0.012–0.038), TOOL_SUBSTITUTION (0.003–0.020), and ACTION_ADDITION (0.005–0.029). The Seq2Seq models are strong only on omissions at the task-end step.

C.2 Scene-Graph Transformer and CUSUM

Scene-graph transformer (SGT). At each step t , SGT encodes the task name, the symbolic scene graph obtained by replaying actions $a_{0:t-1}$ in the PDDL simulator, and the current action a_t ; a causally masked transformer encoder (model width 128, four heads) runs over the steps, and three heads predict the step-level error probability p_t , an auxiliary prefix probability, and the nine type probabilities. Because the graph precedes the action and the action is a separate input, each step can be scored by whether the current action fits the current state. Training uses losses analogous to Seq2Seq (the type loss is applied at error steps), without positive-class weights on the step heads and without truncating trajectories, for three epochs (6,552 steps of eight episodes) with AdamW at learning rate 3×10^{-4} ; the checkpoint with the highest validation M2 AUPRC is kept. We verified causality on the trained model: changing any action after step t changes the outputs at steps $\leq t$ by at most 2.9×10^{-6} (floating-point noise).

CUSUM. A dedicated change-point detector without learning: a task-conditioned action-bigram model $P(a_t \mid a_{t-1})$ with add- k smoothing and unigram back-off is fitted on the correct training episodes; the per-step surprise $s_t = -\log P(a_t \mid a_{t-1})$ drives the CUSUM statistic $S_t = \max(0, S_{t-1} + s_t - k)$, where k is the mean surprise on the training data, and $p_t = S_t / (S_t + k)$. CUSUM makes no type prediction, so its M4 is 0.

C.3 Symbolic Baselines

Both symbolic baselines replay the observed actions in the PDDL simulator and therefore use the domain model, which the learned monitors do not receive.

Precondition checker. An alarm ($p_t = 1$) is raised at any step whose action violates its PDDL preconditions in the replayed state; all other steps receive $p_t = 0$. Because seven of the nine categories pass strict replay by construction (Section A.7), it can fire only on the semantic precondition relaxed for substitutions, and it raises an alarm on fewer than 5% of errorful test episodes.

Re-planner. The re-planner additionally receives the task’s *template-level* formal goal—the subgoals of the task template with object placeholders, e.g. “some bread is toasted and buttered and lies on some plate”—but not the episode’s grounded goal, so it does not know which objects the person will use or the correct final state. It uses the generator’s per-subgoal STRIPS planner. At every (re-)planning point it scores the admissible bindings of the template placeholders by the number of subgoals already satisfied and by the objects the person has touched, plans for the three best bindings, and keeps the shortest plan. It then tracks the plan: an action that matches the next planned action advances it ($p_t = 0$); an action that violates its preconditions raises $p_t = 1$; any other action triggers re-planning from the new state, and if the new plan is $\Delta > 0$ steps longer than the remaining old plan minus one, $p_t = \Delta / (\Delta + 1)$, otherwise $p_t = 0$. At the task-end event, $p_t = 1$ if no admissible binding satisfies all subgoals. When the person walks to a location outside the planner’s task-relevant set, that location is added to the planner’s action space before re-planning. Each episode has a 30-minute budget; 60 test episodes (42 errorful, 18 correct) did not finish and are excluded from M1 and M3, as for any method with missing predictions.

Analysis. The re-planner’s M1 (0.887) comes almost entirely from the end-of-task check: this check alone gives M1 0.802, and with the task-end step removed M1 drops to 0.669 [0.622, 0.714]. On the steps before task end its AUPRC is 0.118, it raises the first alarm early in 84% of detected errorful episodes, and it alarms on 62% of correct episodes: correct executions are not shortest plans, so deviation from a plan is not an error. Its weakest categories are those that leave the goal satisfied, such as spatial misreach and action addition. For reference, the row marked * in Tables 13–15 gives a privileged variant that receives the episode’s grounded goal, i.e. the correct final state (33 test episodes did not finish); it is an upper reference for the re-planner and not a monitor that could be deployed.

Table 13: Main results with 95% confidence intervals (500-episode test set). M2-end: task-end step (omissions); M2-mid: steps before task end (all other errors). Random: M1 0.500, M2 0.036, M2-end 0.274, M2-mid 0.023. *Privileged: receives the episode’s grounded goal.

Method	M1	M2	M2-end	M2-mid
Seq2Seq	0.496 [0.44, 0.55]	0.376 [0.33, 0.43]	0.656 [0.58, 0.73]	0.132 [0.11, 0.17]
Seq2Seq+SG	0.554 [0.50, 0.60]	0.394 [0.34, 0.45]	0.665 [0.58, 0.75]	0.151 [0.13, 0.18]
SGT	0.778 [0.73, 0.82]	0.656 [0.61, 0.70]	0.791 [0.73, 0.85]	0.553 [0.49, 0.62]
CUSUM	0.645 [0.59, 0.69]	0.039 [0.04, 0.04]	0.235 [0.20, 0.28]	0.029 [0.03, 0.03]
Precondition check	0.513 [0.50, 0.52]	0.059 [0.05, 0.08]	0.274 [0.24, 0.31]	0.062 [0.04, 0.09]
Re-planner	0.887 [0.86, 0.91]	0.379 [0.32, 0.45]	0.690 [0.63, 0.76]	0.118 [0.07, 0.17]
Re-planner (episode goal)*	0.922 [0.90, 0.94]	0.371 [0.31, 0.44]	0.673 [0.61, 0.74]	0.125 [0.08, 0.18]

Table 14: M1 and M2-mid by template complexity on the test set (121 simple, 291 medium, 88 complex episodes).

Method	Simple		Medium		Complex	
	M1	M2-mid	M1	M2-mid	M1	M2-mid
Seq2Seq	0.614	0.189	0.498	0.140	0.349	0.047
Seq2Seq+SG	0.706	0.229	0.536	0.152	0.391	0.154
SGT	0.844	0.690	0.797	0.546	0.573	0.406
CUSUM	0.666	0.178	0.654	0.039	0.633	0.031
Precondition check	0.507	0.055	0.516	0.076	0.509	0.039
Re-planner	0.866	0.073	0.897	0.140	0.894	0.150
Re-planner (episode goal)*	0.900	0.098	0.940	0.180	0.923	0.063

Table 15: M3 first-alarm distribution at $\theta=0.5$ on the test set. Missed, early ($\Delta < 0$), on target ($\Delta = 0$), and late ($\Delta > 0$) are shares of the 175 episodes whose first error occurs before task end; false alarm is the share of correct episodes with an alarm. Omissions: share of the 137 omission episodes alarmed at the task-end step / before it. Episodes on which the re-planner timed out are excluded from its rows (145 / 125 / 170 episodes for the template-goal re-planner and 155 / 132 / 180 for the privileged one).

Method	Missed	Early	On target	Late	False alarm	Omission: end / before
Seq2Seq	0.99	0.00	0.00	0.01	0.10	0.46 / 0.00
Seq2Seq+SG	0.98	0.00	0.00	0.02	0.05	0.38 / 0.00
SGT	0.52	0.04	0.38	0.06	0.12	0.58 / 0.01
CUSUM	0.06	0.77	0.07	0.10	0.81	0.01 / 0.80
Precondition check	0.95	0.00	0.05	0.00	0.00	0.00 / 0.00
Re-planner	0.10	0.75	0.05	0.10	0.62	0.33 / 0.66
Re-planner (episode goal)*	0.13	0.74	0.05	0.08	0.33	0.59 / 0.41

C.4 Additional Results

Table 13 gives 95% confidence intervals for the main results (trajectory-level bootstrap stratified by errorful and correct episodes, 1,000 resamples), Table 14 breaks M1 and M2-mid down by template complexity, and Table 15 gives the full first-alarm distribution. The learned monitors degrade with task complexity (SGT: M1 0.844 on simple, 0.573 on complex templates), whereas the re-planner does not, because its end-of-task goal check does not depend on task length.

C.5 LLM Baselines

Protocol. LLMs are evaluated prefix by prefix with the same input as every other monitor. For each evaluated step t of an episode, one request contains the system prompt below and a user message with the task name, the initial scene (every object with its class, location, and initial states), and the actions $a_{0:t}$; the model returns a JSON object for step t only, with `p.instant` (p_t) and nine

cumulative category probabilities (p_{types}). No future action, trajectory length, goal condition, reference execution, or label is included; the task-end event appears only at the final step. On the 60-episode subset this gives 1,201 requests per configuration. Before running, we checked all 1,201 user prompts automatically for these properties.

API settings. All calls use OpenRouter with temperature 0. Because the default reasoning behaviour of the same model differs across providers (e.g. DeepSeek V4 Flash reasons by default on one provider and not on another), each configuration is pinned to a single provider with fallback disabled, and reasoning is set explicitly: disabled, an effort level, or a token budget (Table 16). Requests time out after 120–600 s and are retried twice; steps whose response cannot be parsed are re-requested once with a doubled output limit. All thirteen configurations completed every step. Every response is logged verbatim, together with the reasoning text when the provider returns it, token counts, cost, and the serving provider; the logs are released with the benchmark. The total cost of the thirteen runs was US\$107.

System prompt. Reproduced verbatim (line breaks added for display; an em dash is typeset as “—”).

You are monitoring a human performing a household task in real time, ONE STEP AT A TIME.

You see the task name, the initial scene, and the actions from step 0 up to and including the current step.

You CANNOT see any future action, and you are NOT told how long the task will last.

The last step of every episode is the event ‘human-task-end h’: the person declares the WHOLE task finished.

It appears only when it happens. At that step you can judge whether some step the task needs was never done.

For the CURRENT step, output two judgments:

- p_{instant} (0.0–1.0): probability that an error occurs AT the current step (the current action is where an error happens).
- p_{types} : for each of the 9 error categories, the probability that an error of that category has occurred at or before the current step (cumulative over the episode so far).

IMPORTANT GUIDANCE:

- You are NOT given the formal goal conditions, only the task name. Infer what the task requires from the task name and the scene, as a person watching would.
- By default assume the current step is correct (p_{instant} near 0). Give $p_{\text{instant}} > 0.5$ only with clear evidence that an error happens AT this step.
- A step is correct if it is a reasonable action toward the task, even if it is roundabout or inefficient. Harmless redundancy is NOT an error: walking back and forth between places the task uses, picking something up and putting it back, or repeating an action immediately when the repetition has no additional effect.
- p_{instant} marks only the step where an error happens, not later steps that are merely its consequences.
- p_{types} is cumulative: once an error of some category has happened, keep that category high at all later steps, including the final task-end step.

Error categories (the step in brackets is where p_{instant} should be high):

- OMISSION: a step the task requires was never performed. This can only be confirmed when the person declares the task finished. [the ‘human-task-end’ step]
- SEQUENCE: two steps that must happen in a specific order were done in the wrong order, e.g. applying butter to bread before toasting it. [the step done out of order]
- ANTICIPATION: an item is stored, served or packed before it receives the processing the task requires. [the step that commits the unprocessed item: putting it into its final place, closing the container that holds it, or delivering that container to its final place]
- PERSEVERATION: after doing something else in between, the person repeats an

- action that should be done only once, e.g. adds sugar to the coffee, butters the toast, then adds sugar to the coffee again. [the repeated action]
- OBJECT_SUBSTITUTION: a similar object of the same kind is used in place of the object the task needs. [the first step that uses the substitute]
- TOOL_SUBSTITUTION: a similar tool of the same kind is used in place of the appropriate tool. [the first step that uses the substitute tool]
- ACTION_ADDITION: an item unrelated to the task is put into or onto the task’s target container, plate or tray. Removing it later does not undo the error. [the step that puts it there]
- CAPTURE_ERROR: an action the task needs is applied to an unrelated object of a different kind instead of the task’s object. [the step that acts on the unrelated object]
- SPATIAL_MISREACH: the person goes to a storage location that has nothing to do with the task, opens it, and leaves without taking anything. [the step of leaving]

OUTPUT FORMAT -- respond with ONLY valid JSON (no markdown, no commentary):

```
{
  "step": <int>,
  "p_instant": <float>,
  "p_types": {
    "OMISSION": <float>,
    "SEQUENCE": <float>,
    "ANTICIPATION": <float>,
    "PERSEVERATION": <float>,
    "OBJECT_SUBSTITUTION": <float>,
    "TOOL_SUBSTITUTION": <float>,
    "ACTION_ADDITION": <float>,
    "CAPTURE_ERROR": <float>,
    "SPATIAL_MISREACH": <float>
  }
}
```

Output exactly ONE such object, for the current step only.

User prompt template.

Task: <task name, underscores replaced by spaces>

Initial scene: <object>:<CLASS>@<location>[<states>] ...

Actions observed so far:

Step 0: <action> h <arg1> <arg2> ...

...

Step t: <action> h <arg1> ...

Judge the current step (Step t). Output the JSON object for step t only.

C.6 Full LLM Results

Table 17 reports all thirteen configurations on the 60-episode subset (30 errorful, 30 correct; 1,201 evaluated steps, 33 of them error steps; M2 random 0.027, M2-end 0.067, M2-mid 0.025), with 95% bootstrap intervals, together with the other baselines re-scored on the same episodes; Table 18 gives the first-alarm distribution.

Key observations.

1. **Models cannot be ranked on M1.** With 60 episodes, the M1 intervals of all configurations overlap. The best upper bounds stay below the human reference on both M1 (0.837 vs. 0.967) and M2 (0.482 vs. 0.748).
2. **Reasoning matters for localization and diagnosis.** With reasoning enabled, the four larger models reach M2-mid 0.18–0.48 and M4 0.43–0.52; with reasoning disabled, M2-mid is at most 0.15 and M4 at most 0.31, and Qwen3.6 and DeepSeek V4 Flash output near-zero scores at almost every step. At the steps before task end, the best reasoning configurations (DeepSeek V4 Pro 0.478, Claude Sonnet 4.6 0.451) are on par with SGT on the same episodes (0.455); none of them confirms omissions at task end well (M2-end at most 0.22).
3. **More effort does not help further.** GPT-5.5 low and medium are indistinguishable on every metric (M2 0.238 vs. 0.231), at 65% higher cost for medium.
4. **Reasoning trades misses for false alarms.** The reasoning configurations give the most on-target first alarms (42–50% of errorful episodes) but alarm on 30–77% of correct episodes; without reasoning, most models rarely alarm at all.

Table 16: LLM configurations. Each configuration is pinned to one OpenRouter provider (fallback disabled); reasoning is set explicitly. ^aReports one reasoning token per request with reasoning disabled, and no reasoning text. ^bThe provider rejects requests with reasoning disabled (“reasoning is mandatory”), so the lowest effort is used.

Model	OpenRouter ID	Provider	Reasoning	Max tokens
Qwen3.6-35B-A3B	qwen/qwen3.6-35b-a3b	DeepInfra (fp8)	disabled	512
Gemini 3.1 Flash Lite	google/gemini-3.1-flash-lite	Google AI Studio	disabled	512
Kimi K2.6	moonshotai/kimi-k2.6	Moonshot AI	disabled ^a	512
Grok 4.3	x-ai/grok-4.3	xAI	disabled	512
GLM-5.1	z-ai/glm-5.1	Z.AI (fp8)	disabled	512
DeepSeek V4 Flash	deepseek/deepseek-v4-flash	Novita (fp8)	disabled	512
GPT-5.4 mini	openai/gpt-5.4-mini	OpenAI	disabled	512
MiniMax-M2.7	minimax/minimax-m2.7	MiniMax	low ^b	4,000
DeepSeek V4 Pro	deepseek/deepseek-v4-pro	Alibaba (fp8)	medium	16,000
Gemini 3.1 Pro	google/gemini-3.1-pro-preview	Google AI Studio	medium	8,000
Claude Sonnet 4.6	anthropic/claude-sonnet-4.6	Anthropic	2,000 tokens	4,000
GPT-5.5	openai/gpt-5.5	OpenAI	low	8,000
GPT-5.5	openai/gpt-5.5	OpenAI	medium	8,000

Table 17: LLM results on the 60-episode subset, sorted by M1, with the other baselines re-scored on the same episodes. M4 includes correct episodes (Section C.1). 95% intervals from a trajectory-level bootstrap stratified by errorful and correct episodes (1,000 resamples).

Model	M1	M2	M2-end	M2-mid	M4
Kimi K2.6	0.718 [0.60, 0.84]	0.140 [0.06, 0.25]	0.067	0.154	0.179
Claude Sonnet 4.6 (2,000 tokens)	0.697 [0.56, 0.82]	0.325 [0.20, 0.48]	0.111	0.451	0.432
DeepSeek V4 Pro (medium)	0.679 [0.55, 0.81]	0.286 [0.17, 0.44]	0.218	0.478	0.480
GPT-5.5 (low)	0.631 [0.48, 0.77]	0.238 [0.14, 0.38]	0.133	0.394	0.489
GPT-5.5 (medium)	0.617 [0.47, 0.75]	0.231 [0.15, 0.37]	0.116	0.430	0.477
Grok 4.3	0.584 [0.48, 0.69]	0.086 [0.03, 0.18]	0.067	0.093	0.107
GPT-5.4 mini	0.579 [0.45, 0.71]	0.055 [0.03, 0.12]	0.067	0.059	0.143
GLM-5.1	0.576 [0.45, 0.72]	0.077 [0.05, 0.13]	0.081	0.114	0.305
Gemini 3.1 Pro (medium)	0.572 [0.44, 0.69]	0.119 [0.07, 0.20]	0.069	0.175	0.519
Gemini 3.1 Flash Lite	0.564 [0.49, 0.65]	0.061 [0.03, 0.16]	0.067	0.064	0.086
MiniMax-M2.7 (low)	0.527 [0.38, 0.67]	0.079 [0.04, 0.17]	0.113	0.090	0.192
Qwen3.6-35B-A3B	0.517 [0.50, 0.55]	0.057 [0.02, 0.12]	0.067	0.059	0.109
DeepSeek V4 Flash	0.500 [0.50, 0.50]	0.027 [0.02, 0.03]	0.067	0.025	0.099
Seq2Seq	0.336 [0.20, 0.48]	0.107 [0.05, 0.21]	0.375	0.085	0.452
Seq2Seq+SG	0.350 [0.20, 0.50]	0.114 [0.07, 0.22]	0.277	0.112	0.533
SGT	0.683 [0.53, 0.80]	0.423 [0.29, 0.59]	0.458	0.455	0.490
CUSUM	0.692 [0.56, 0.82]	0.036 [0.03, 0.05]	0.119	0.035	0.000
Precondition check	0.517 [0.50, 0.55]	0.057 [0.02, 0.12]	0.067	0.059	0.000
Re-planner	0.871 [0.76, 0.96]	0.113 [0.05, 0.26]	0.273	0.158	0.084

Table 18: M3 first-alarm distribution at $\theta=0.5$ on the 60-episode subset: shares of the 26 errorful episodes whose first error occurs before task end (missed, early, on target, late) and false-alarm rate on the 30 correct episodes; for the re-planner, 22 and 28 episodes, excluding timeouts.

Model	Missed	Early	On target	Late	False alarm
Kimi K2.6	0.46	0.38	0.12	0.04	0.27
Claude Sonnet 4.6 (2,000 tokens)	0.31	0.08	0.50	0.12	0.30
DeepSeek V4 Pro (medium)	0.27	0.19	0.42	0.12	0.47
GPT-5.5 (low)	0.15	0.08	0.46	0.31	0.57
GPT-5.5 (medium)	0.23	0.12	0.46	0.19	0.50
Grok 4.3	0.69	0.27	0.04	0.00	0.13
GPT-5.4 mini	0.73	0.19	0.04	0.04	0.07
GLM-5.1	0.19	0.27	0.15	0.38	0.67
Gemini 3.1 Pro (medium)	0.08	0.23	0.42	0.27	0.77
Gemini 3.1 Flash Lite	0.81	0.08	0.08	0.04	0.03
MiniMax-M2.7 (low)	0.54	0.23	0.15	0.08	0.47
Qwen3.6-35B-A3B	0.96	0.00	0.04	0.00	0.00
DeepSeek V4 Flash	1.00	0.00	0.00	0.00	0.00
Seq2Seq	1.00	0.00	0.00	0.00	0.07
Seq2Seq+SG	1.00	0.00	0.00	0.00	0.07
SGT	0.73	0.04	0.23	0.00	0.07
CUSUM	0.04	0.85	0.04	0.08	0.87
Precondition check	0.96	0.00	0.04	0.00	0.00
Re-planner	0.09	0.82	0.00	0.09	0.64

C.7 Expert Review

To obtain an initial check of generator fidelity, two domain experts (neuropsychology; nursing care) jointly reviewed 24 randomly selected NAT-aligned episodes (12 errorful, 12 correct) without access to the labels. They judged 10 of the 12 injected errors plausible, identified 11 of the 12 correct episodes as correct, and matched the generator’s error category on 8 of the 12 errorful episodes. The experts noted that the NAT alignment yields realistic tasks, whereas the plausibility of an error depends on the scene, which is consistent with the scene-specific feasibility rules of Section A.6. Mapping the generated errors to NAT error instances gives three direct and four structural matches. These results provide initial expert support for the generator, but they are not a patient-level validation; results of further expert validation will be released on the project website.