SSPEC: READING THE ASK-OR-PROCEED DECISION OFF THE DECLARED INTERFACE OF AN AGENT TOOL CALL

Anonymous authors
Paper under double-blind review

ABSTRACT

On a published travel-planning agent, 0.8963 of call arguments are not written by the executor itself, yet actions that cannot be undone consume them. However, ask-or-proceed instruments question the executor at the step that writes a value, and the executor reports only what it knew it chose. We propose Schema-Sourced Specification (SSPEC), a framework that reads the ask-or-proceed decision off the declared interface. A recovery operator enumerates the fields a call puts in force from the declared schema and returns those no witness in the evidence buffer carries. An exposure split defers the question on each such value to the first unrevertable dispatch that consumes it. Budgeted selection spends an interruption allowance on the groups of fields that one declared unit or frame parameter closes. The key insight is that the interface already carries what the executor cannot report: the schema lists the values in force, and the dispatch fixes when each one matters. SSPEC reads 92.60% API match rate on default-filled-only episodes against 83.70% for the strongest published predictor and recovers 68.4% of the unsupported values there against 38.6% for executor self-report. On ProClare it asks 1.6657 questions per episode against 5.0000 for an every-turn rule and completes 49.5000% of tasks against 47.0000% for the strongest published policy. On the Minecraft Dialogue Corpus, withdrawing the exposure split raises irreversible exposure from 8.6100% to 17.2500% at an almost unchanged question count, and withdrawing budgeted selection raises questions from 0.9320 to 1.1890 at an almost unchanged exposure; the two costs are charged at different places.

1 INTRODUCTION

An agent that calls a declared tool must put a value in every parameter of that call, whether or not anyone decided it (Zhang et al., 2024; Shi et al., 2022; Ahmadian et al., 2026). Where the instruction leaves a field open, the value still arrives from somewhere: the executor writes one, an earlier call leaves one behind, or the interface supplies its declared default. The call validates in all three cases, and nothing in the emitted arguments marks which of them a user actually chose. On the tool-use episodes of a published travel-planning agent (Zhang et al., 2024), 0.8963 of call arguments are not written by the executor itself. Actions with real effects then consume those arguments, among them a confirmed booking, an issued order and a placed block; what an agent asks before it acts therefore decides which of the values it fixed a user ever sees (Ye et al., 2025; Shi et al., 2022).

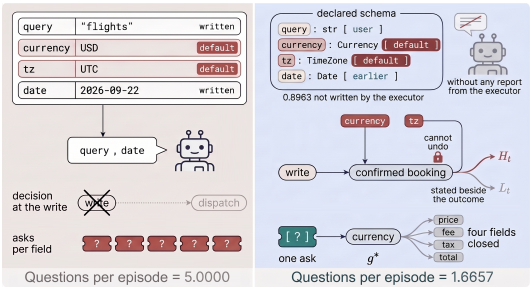

Figure 1: **Left:** the executor names only the fields it wrote; two default-filled values go unmentioned, and each field costs a question. **Right:** SSPEC recovers them from the declared schema and closes four fields with one ask.

One family of interaction policies reads the instruction and the tool documentation for the information an executor still needs, and asks when a required argument cannot be filled (Zhang et al.,

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

2024; Ahmadian et al., 2026). A second decides between asking and executing from the dialogue that precedes the step, training a turn-level policy to recognise the turns on which a question pays for itself (Shi et al., 2022). A third asks on every turn, treating the interruption as cheap and the user as available (Ye et al., 2025; Montazeralghaem et al., 2025). Together the three have made the ask-or-proceed decision measurable, with annotated unsettled arguments, shared corpora and reported clarification accuracy (Zhang et al., 2024; Shi et al., 2022; Ye et al., 2025). The same decision now carries embodied help-seeking and interactive data wrangling, where one question recovers an intent the interface never received (Ramrakhya et al., 2025; Chen et al., 2025). Despite these advances, all three take an account of what the call still lacks as the carrier on which a value becomes visible.

However, a value the executor never represented as a choice is not on that carrier at all: an argument the interface filled from its declared default is present, well typed and unmentioned, and a test for missing information and a self-report each reach only some of these arguments. What such a value costs is settled elsewhere as well, because the cost belongs to the actions it later feeds, and the first action that cannot be undone is rarely the step that wrote it. The questions that would settle it are both scarce and overlapping, since declared types carry unit and frame parameters under which one answer closes a group of fields that a field-by-field policy pays for one at a time. Yet these three gaps share a more fundamental limitation: what an ask-or-proceed instrument can recover is bounded by the carrier it questions rather than by the accuracy it attains.

To address these limitations, we propose SSPEC, a framework that recovers the values an execution fixed without support from the declared tool interface and spends a bounded interruption allowance on them at the consuming dispatch. The key insight is that the interface already carries what the executor cannot report, since the schema domain enumerates the fields a call puts in force, the declared defaults say which of them nobody wrote, and the evidence buffer decides which are grounded. The two decisions a policy has to make are then two readings of one execution taken at two moments, the values in force read off the declared interface and their urgency read off the dispatch about to issue. SSPEC has three components: a recovery operator that returns the fields in force without a witness by taking the complement of the emitted assignment inside the declared schema; an exposure split that sorts that set at the dispatch about to issue rather than at the step that wrote the value; and a budgeted selection that spends what remains of the allowance on the groups one declared unit or frame parameter closes at once. Figure 3 lays out how the three components compose over a single step. Our contributions are as follows:

- We identify the carrier gap behind this decision: the published instruments that question the executor range over the values the executor was aware of choosing.
- We propose SSPEC, which reads the values in force off the declared schema and the evidence buffer, puts the question at the dispatch before an irreversible effect rather than at the step that wrote the value, and spends the interruption allowance on the groups one declared parameter closes.
- Our framework reads 92.60 API match rate on the default-filled episodes of Ask-before-Plan against 83.70 for the strongest published predictor, and asks 1.6657 questions per episode on ProClare against 5.0000 for an every-turn-ask policy.

2 MOTIVATION

To understand why asking the executor leaves default-filled arguments unprotected, we examine where each value is fixed and what consumes it, revealing three challenges.

Challenge #1: Candidates the executor cannot report. Published instruments for this decision question the executor (Zhang et al., 2024; Shi et al., 2022). Figure 2(left) reads three instruments on two axes at once: reported accuracy climbs from 22.6000 to 76.8000 while coverage of the annotated unsettled set stays between 0.0565 and 0.3458, and the most accurate instrument is not the one with the widest coverage. A self-report ranges over the executor's representation of what it needed; a field never represented as a choice is absent by construction rather than by error. A candidate set has to be recovered from what the call declares, not from what the executor reports.

Challenge #2: Cost that appears after writing. A policy that decides when a value is written treats its cost as a property of the writing step (Shi et al., 2022; Ahmadian et al., 2026). Fig-

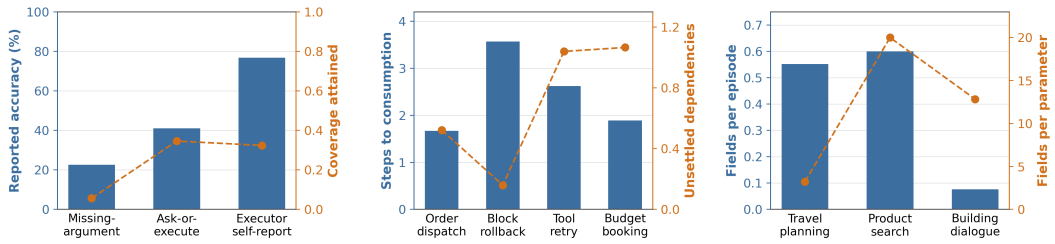

Figure 2: **(Left)** Accuracy rises monotonically while coverage does not. **(Middle)** Consumption trails writing in every case. **(Right)** One parameter closes a whole group. Details in Appendix A.2.

ure 2(middle) places each published case at both positions. The distance from fixing to first unrevertable consumption is strictly positive everywhere, from 1.67 steps to 3.57, and the cases carrying the most unsettled dependencies are not the ones with the longest distance. At the writing step the consuming actions have not been chosen, leaving urgency undefined there, where no care can recover it; it becomes defined only at the dispatch, the position no published policy re-reads. The decision therefore belongs at the dispatch, where that cost first exists.

Challenge #3: A budget spent on groups. Asking whenever a field looks open spends an interruption per field and assumes the user tolerates that count (Ye et al., 2025). Figure 2(right) sets the number of separately underspecified fields per episode against the number of fields one declared unit or frame parameter closes at once. The two differ by factors of 5.8, 33 and 169 across the three carriers, reaching 0.6000 against 20.00 on product search. Fields share the declared parameters their types carry rather than being independent slots, and a field-by-field policy pays an interruption for each member of a group one question would have closed. What a question should be aimed at is the declared parameter that reaches a group, which a single question then settles in full.

3 Problem Formulation

To address the three challenges of Section 2, we cast the ask-or-proceed decision as a constrained expected-utility maximisation under an interruption budget and an irreversibility constraint,

$$\max_{\pi \in \Pi} \mathbb{E}\big[U(\pi)\big] \quad \text{s.t.} \quad Q(\pi) \le b, \qquad \mathrm{dep}(a_t) \cap S_t = \emptyset \ \text{ whenever } \ \mathrm{irrev}(a_t) \tag{1}$$

where a policy $\pi \in \Pi$ either issues the call a_t it faces or puts one question to the user, U measures how well the delivered artefact matches what the user wanted, Q counts the interruptions it imposes at one unit per question whatever that question settles, $b \in \mathbb{N}$ is the allowance granted for one task, and $\mathrm{dep}(a_t)$ and $\mathrm{irrev}(a_t)$ name the fields the call consumes and an effect it cannot undo. The set S_t collects the fields the interface declares for a_t whose value at dispatch is grounded by nothing the episode has committed, whoever supplied it. The recovery operator estimates this set.

Three features of this program answer those challenges. Defining S_t over the fields the declared interface fixes, which admits a value the executor never emitted, answers Challenge #1. Binding the constraint to $\mathrm{dep}(a_t)$ at the consuming call rather than at the write answers Challenge #2. Charging Q one unit per question of any reach answers Challenge #3.

4 Method

4.1 SSPEC Overview

SSPEC resolves the ask-or-proceed decision of Eq. 1 for a single tool call. As illustrated in Figure 3, it comprises three components: a recovery operator that returns the fields a call puts in force without support, an exposure split that sorts them against the call about to issue, and budgeted selection that spends what remains of the allowance. Each component reads the output of the one before it.

The three are one integrated pass, because each consumes what the one before it produced: exposure cannot classify a field the recovery operator did not return, and the allowance cannot be aimed before

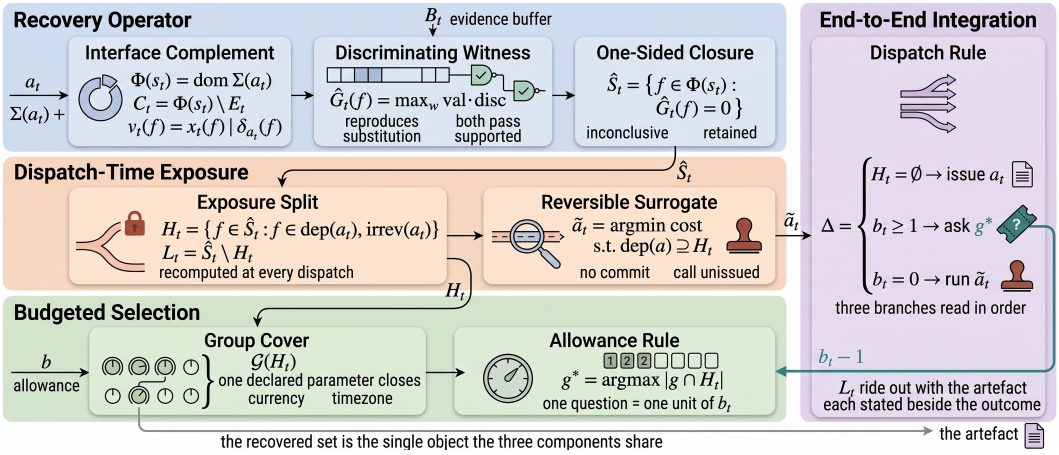

Figure 3: SSPEC over one step of an episode. **(Top)** the recovery operator returns the fields in force without a witness. **(Middle)** the exposure split sorts them at the issuing dispatch. **(Bottom)** selection charges one unit per group. **(Right)** the dispatch rule in three branches.

the exposed part is known. The order is therefore forced by the dependency. What the composition buys is a decision read off the declaration of a call rather than off the executor's account of it, with the allowance spent only where an effect cannot be undone.

4.2 Recovery Operator

To address Challenge #1, we propose the recovery operator, the top band of Figure 3, which supplies the set S_t of Eq. 1. The fields the schema puts in force are first enumerated, then tested against the evidence buffer, and finally the inconclusive ones are retained.

Interface Complement. The input is the imminent action a_t with its interface schema $\Sigma(a_t)$ and the assignment x_t the call emits, whether the executor wrote it out or carried it over from an earlier call; we compute the complement of the emitted fields inside the schema domain and read the value in force on each field from the interface,

$$\Phi(s_t) = \mathrm{dom}\,\Sigma(a_t), \qquad C_t = \Phi(s_t) \setminus E_t, \qquad v_t(f) = \begin{cases} x_t(f), & f \in E_t \\ \delta_{a_t}(f), & f \in C_t \end{cases} \tag{2}$$

where the state s_t pairs the executed prefix with the imminent call a_t, $\Phi(s_t)$ is the field domain the declared schema puts in force, C_t collects the fields no emitted assignment mentions, E_t is the set of fields the call emits, whether the executor wrote the value out or carried it over from an earlier call, and $\delta_{a_t}(f)$ is the value the interface declares as that field's default. The output is the pair of every field of the domain with the value in force on it and a label recording which of the two sources supplied it. The label marks every default-filled field before any acceptance test runs.

Enumeration over $\Phi(s_t)$ rather than over E_t is what makes a default-filled argument a candidate at all. A field in C_t carries a value that the call will dispatch and that validates against the schema; a missing-argument test rarely fires on it, because nothing is missing. The artefact this subsection hands on is a labelled assignment over the whole domain, on which the acceptance test of the next block can be run field by field. No field of the domain is dropped at this stage.

Discriminating Witness. A candidate witness w is a span of B_t together with one typed read of that span; we apply two tests to it, reproduction of the value in force and sensitivity to a type-preserving substitution on the span itself, and accept the field as supported when one witness passes both,

$$\widehat{G}_t(f) = \max_{w \in \mathcal{W}(B_t)} \mathrm{val}_t(f, w) \cdot \mathrm{disc}_t(f, w) \tag{3}$$

where $\widehat{G}_t(f)$ is the support indicator of field f, B_t is the immutable evidence buffer the executed prefix has committed, $\mathcal{W}(B_t)$ enumerates the candidate witnesses over it, $\mathrm{val}_t(f, w)$ records whether

w reproduces $v_t(f)$, and $\mathrm{disc}_t(f, w)$ records whether replacing the span with a type-preserving alternative changes what w returns. The product is one when a single witness satisfies both tests and zero otherwise, including when $\mathcal{W}(B_t)$ is empty.

The second test is what separates a span that carries the value from a span that merely contains it. A date that appears in the buffer for an unrelated reason reproduces a datetime argument by coincidence, and the reproduction test alone admits it; substituting a different date on that span leaves the extracted value unchanged; the discrimination test rejects it. Acceptance therefore admits fewer witnesses than reproduction alone, and the fields it declines are drawn from the ones whose grounding was coincidental. A declined field stays a candidate for the closure that follows.

One-Sided Closure. We select the unsupported set as the fields of the domain that no witness carried,

$$\widehat{S}_t = \{\, f \in \Phi(s_t) : \widehat{G}_t(f) = 0 \,\}, \qquad R\big(\Phi(s_t), B_t\big) = \widehat{S}_t \tag{4}$$

where $\widehat{S}_t$ is the recovered estimate of the unsupported set S_t, carried into the exposure stage, $\widehat{G}_t$ is the support indicator of Eq. 3, and R denotes the recovery operator these three blocks assemble. Dropping a supported field into $\widehat{S}_t$ costs at most one question, while dropping an unsupported field out of it removes the only chance to catch the value, and the closure therefore keeps the field whenever the witness evidence is inconclusive.

The asymmetry is what makes the closure usable on a real schema. An episode whose evidence buffer is thin returns an inconclusive indicator on many fields at once, and a rule that resolved those fields out of the set would hand the exposure stage a candidate set smaller than the one the episode actually fixed without support. We then pay for the retained fields in questions charged against the allowance, and the exposure split narrows the retained set again before any of them is charged. Eq. 4 instantiates the set S_t of Eq. 1, recovered from the declared schema and the evidence buffer without any report from the executor. Its output and the call about to issue are what the exposure split reads.

4.3 Dispatch-Time Exposure

To address Challenge #2, we propose dispatch-time exposure, the middle band of Figure 3, which enforces the irreversibility constraint of Eq. 1 on the recovered set at the consuming call. The recovered set is first split by effect class, and a reversible surrogate then stands in once the allowance is spent. A field that no unrevertable call consumes is never charged against the allowance.

Exposure Split. The input is $\widehat{S}_t$ and the call about to be issued; we then sort each recovered field by the effect class of the action that consumes it, separating a field feeding an effect the episode cannot undo from one whose consequence is a stated value in the delivered artefact,

$$H_t = \{\, f \in \widehat{S}_t : f \in \mathrm{dep}(a_t),\ \mathrm{irrev}(a_t) \,\}, \qquad L_t = \widehat{S}_t \setminus H_t \tag{5}$$

where $\mathrm{dep}(a_t)$ collects the fields the imminent call consumes, $\mathrm{irrev}(a_t)$ marks its effect class as unrevertable, H_t is the hard-exposed part and L_t the light-exposed part of the step. The same field moves between the two parts as the episode proceeds, because the split is recomputed at every dispatch. A value consumed only by reversible calls therefore never enters H_t.

Recomputation is the whole content of the choice. A value written early and consumed late sits in L_t for the steps in between and enters H_t only at the call that spends it, which is the first moment at which the cost of having guessed it is a property of the trajectory rather than a prediction about it. A policy that fixed the decision at the write would have had to make that prediction with the consuming action not yet chosen. At the dispatch its effect class is read directly from the interface.

Reversible Surrogate. Given a non-empty H_t and the imminent call, we select a reversible action whose observable outcome bounds the same decision, run it in place of the unrevertable one, and require an exact commit before the effect is taken,

$$\tilde{a}_t = \arg\min_{a \in \mathcal{A}_r(a_t)} \mathrm{cost}(a) \quad \text{s.t.} \quad \mathrm{dep}(a) \supseteq H_t \tag{6}$$

where $\mathcal{A}_r(a_t)$ is the set of reversible actions the interface offers in place of a_t, $\mathrm{cost}(a)$ is the step cost of running one, and the constraint requires the surrogate to depend on the hard-exposed fields it

stands in for. A surrogate that returns without an exact commit leaves the unrevertable call unissued rather than issuing it on the surrogate's evidence.

Requiring the exact commit is what keeps the surrogate from becoming a cheaper guess. A reversible probe narrows the hard-exposed fields it depends on by observation, and the episode proceeds only when that observation fixes each of them to a single value the schema admits; an observation that merely makes one value likelier leaves the field in H_t and the unrevertable call unissued. The artefact this subsection hands to the dispatch rule is therefore either a settled field or an unspent step, never a value the surrogate inferred. Eq. 5 and Eq. 6 instantiate the irreversibility constraint of Eq. 1, evaluated at the call that consumes a value rather than at the step that wrote it.

4.4 Budgeted Selection

To address Challenge #3, we propose budgeted selection, the bottom band of Figure 3, which spends the allowance b of Eq. 1 on groups rather than on fields. The forced set is first covered by declared parameter groups, and the widest group is then charged one unit.

Group Cover. The input is the hard-exposed set H_t and b_t, what remains of the allowance b at this step. Fields inherit the unit and frame parameters of their declared types; a single question about a currency or a timezone settles every field derived from it, and $\mathcal{G}(H_t)$ enumerates the groups a single declared parameter closes, a field that no declared parameter reaches forming a group on its own. A cover of H_t by members of $\mathcal{G}(H_t)$ is a schedule of questions that settles the forced set, while a ranking over individual fields spends one unit per field to reach the same state.

Allowance Rule. Each question ranges over one group, taken as the group closing the largest share of what is still forced,

$$g^\star = \arg\max_{g \in \mathcal{G}(H_t)} \left| g \cap H_t \right| \tag{7}$$

where $g^\star$ is the group the next question ranges over. One question costs one unit of b_t whatever the size of the group it closes. The forced set is re-read after the answer; a cover of H_t is completed over successive dispatches rather than chosen in one.

Selecting over groups keeps the allowance usable when the coupling is strong. On a graph in which no node admits an alternative derivation the greedy cover attains the minimum number of questions; where alternative derivations exist it can stand above that minimum. Eq. 7 instantiates the allowance constraint $Q(\pi) \le b$ of Eq. 1, one unit per group. Details in Appendix A.6.

4.5 End-to-End Integration

Dispatch Rule. Composing the recovery operator, the exposure split and the allowance rule, as the right column of Figure 3 shows, the policy of Eq. 1 becomes a single rule read at every dispatch,

$$\Delta = \begin{cases} \text{issue } a_t, & H_t = \emptyset \\ \text{ask } g^\star, & H_t \neq \emptyset,\ b_t \geq 1 \\ \text{run } \tilde{a}_t, & H_t \neq \emptyset,\ b_t = 0 \end{cases} \tag{8}$$

where Δ is the dispatch rule that instantiates the policy π of Eq. 1, the three branches are read in order, and b_t decreases by one unit each time the second branch is taken. Fields in L_t never reach this rule: they ride out with the artefact, each stated beside the outcome as a value the delivery fixed without support. The complete procedure is summarized in Algorithm 1 of Appendix A.7.

Framework Summary. SSPEC reads the decision off the declared interface: a value the executor never reported and the interface supplied from its default can be asked about on the same footing as an argument the user left out. The allowance is spent only where an effect cannot be undone.

Table 1: Ask-or-proceed on the default-filled-only episodes of Ask-before-Plan, in percent.

Policy	API match rate ↑	Final pass rate ↑
No-ask executor	–	0.00
Executor self-report	4.70	0.10
Clarification predictor (Zhang et al., 2024)	83.70	0.10
Ask-or-execute (Shi et al., 2022)	0.70	0.00
Every-turn ask (Ye et al., 2025)	0.50	0.00
SSPEC (ours)	**92.60**	**0.60**

5 EXPERIMENTS

5.1 IMPLEMENTATION DETAILS

Datasets. The carriers are Ask-before-Plan travel planning (Zhang et al., 2024), the Minecraft Dialogue Corpus (Shi et al., 2022), and ProClare product search (Ye et al., 2025). Ask-before-Plan is the primary carrier: it annotates unsettled arguments and declares a schema with defaults. Episodes fall into four regimes by how settled the imminent call's arguments are: fully specified, one unsupported value, several unsupported values and default filled only. ProClare supplies the cost comparison and the Minecraft Dialogue Corpus the online ablation. Details in Appendix A.8.

Baselines. The compared policies are the clarification predictor of Zhang et al. (2024), the ask-or-execute policy of Shi et al. (2022), the every-turn-ask policy of Ye et al. (2025), an executor self-report, and a no-ask executor. The first three are the published instruments; the last two bound the comparison from the self-report side and from asking nothing. The clarification predictor decides from the dialogue state before the call is composed, ask-or-execute decides at the turn level, and every-turn ask asks at every turn whatever the episode has settled. Details in Appendix A.9.

Evaluation metrics. Clarification accuracy, API match rate and final pass rate are percentages, higher is better (Zhang et al., 2024). Three further metrics are unsupported-value recall, irreversible exposure rate and interruptions per episode. Unsupported-value recall is the share of unsupported values an instrument flags, irreversible exposure the share of unrevertable calls that consume such a value, and interruptions per episode the questions asked. Details in Appendix A.10.

Hyperparameters. Every SSPEC configuration is read at one fixed interruption allowance b, and equal-cost covers are broken by one fixed rule. The witness family enumerates the committed spans of the evidence buffer: a witness is never a generated paraphrase. The effect class $\mathrm{irrev}(a_t)$ is read from the carrier's declaration rather than inferred from the action name. The substitution that tests discrimination replaces a span with an alternative of the same declared type, and episodes whose effect declaration is wrong are reported as a separate failure mode. Details in Appendix A.11.

5.2 MAIN RESULTS

Effectiveness on Default-Filled Arguments. Table 1 reports the six policies on the regime in which every unsettled argument was supplied by the interface from its declared default. SSPEC reaches 92.60 API match rate against 83.70 for the strongest published predictor and 4.70 for executor self-report, and 0.60 final pass rate against 0.10. The gap is attributable to where the candidate set comes from: the interface complement of Eq. 2 enumerates those arguments, and an instrument that questions the executor reaches only part of them. Details in Appendix A.4.

Question Cost and Task Completion. Table 2 reports what each policy charges the user and what the episode delivers on a carrier whose actions include unrevertable dispatches. SSPEC asks 1.6657 questions per episode against 3.1455 for the clarification predictor, and task completion does not pay for that: it reads 49.5000 against 47.0000 for the strongest published policy on this carrier. Completion does not track the number of questions, since the no-ask executor asks

Table 2: Question cost, task completion and irreversible exposure on ProClare, completion and exposure in percent.

Policy	Questions ↓	Completion ↑	Exposure ↓
No-ask executor	**0.0000**	34.8000	34.2000
Clarification predictor	3.1455	47.0000	22.0000
Ask-or-execute	1.9150	39.5000	29.5000
Every-turn ask	5.0000	45.0000	24.0000
SSPEC (ours)	1.6657	**49.5000**	**12.4000**

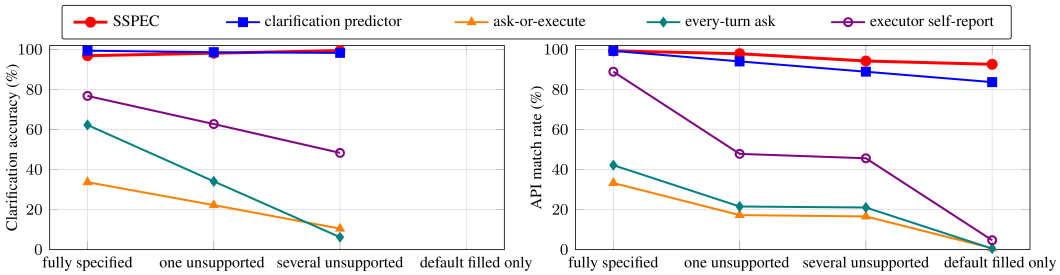

Figure 4: **(Left)** Clarification accuracy across the first three specification regimes, ordered on the horizontal axis from fully specified to default filled only. **(Right)** API match rate across all four, in percent. Bold red: SSPEC. More details are provided in Appendix A.4.

Table 3: Four ask-or-proceed instruments by episode regime; recall in percent, asks per episode.

Instrument	Recall, one ↑	Recall, default ↑	Asks, full ↓	Asks, one ↓	Asks, default ↓
SSPEC (ours)	82.1	68.4	0.0302	0.3054	0.3138
Executor self-report	48.5	38.6	0.6359	0.6418	0.6499
Missing-argument trigger	41.9	17.7	0.3339	0.3356	0.3674
Every-turn ask	99.4	99.4	5.0000	5.0000	5.0000

nothing and completes least at 34.8000 while the every-turn rule asks the most and completes below SSPEC. The ask-or-execute policy asks a comparable 1.9150 questions and completes 39.5000, ten points below SSPEC. Irreversible exposure falls to 12.4000 under SSPEC against 22.0000 for the clarification predictor and 34.2000 for the no-ask executor. What separates them is where the questions land, because the allowance is spent only where the consuming action cannot be undone, by the split of Eq. 5. Details in Appendix A.4.

Specification Regimes. Figure 4 follows the policies across the four specification regimes. In the left panel SSPEC and the clarification predictor stay close in clarification accuracy over the first three regimes, while in the right panel their API match rates separate as the regimes move toward default filled only, ending at 92.60 and 83.70. Deciding correctly whether to ask does not by itself settle the arguments of the call; which values the candidate set contains does, which is the carrier gap of Challenge #1. Executor self-report falls to 4.70 API match rate on the default-filled regime.

Reach of the Questioned Carrier. Table 3 reads four instruments, which separates what an instrument can reach from how often it asks. Self-report recall falls from 48.5% with one unsupported value to 38.6% once the unsettled values are default-filled, and the missing-argument trigger falls further, to 17.7%, while the framework reads 68.4% there at 0.3138 asks, against 99.4% at 5.0000 asks for the every-turn rule. On fully specified episodes, where no question is needed, SSPEC asks 0.0302 times per episode against 0.6359 for self-report, which asks more in every regime and recovers less in both regimes with reported recall. Details in Appendix A.5.

5.3 Ablation Study and Analysis

Effectiveness of the Online Components. Table 5 switches the three online components on and off on the Minecraft Dialogue Corpus, and they separate cleanly onto the two quantities. Removing the exposure split raises the exposure rate from 8.6100 to 17.2500 at an almost unchanged question count; removing the reversible surrogate, so that a verification question precedes every unrevertable call, raises questions per episode to 1.9730, and removing budgeted selection raises them to 1.1890, against 0.9320 with

Table 5: Online components switched on and off on the Minecraft Dialogue Corpus, exposure in percent.

Configuration	Exposure ↓	Questions ↓
All three removed	17.4400	2.2915
w/o Exposure Split	17.2500	0.9935
Surrogate and selection removed	8.8000	2.2300
w/o Reversible Surrogate	8.7600	1.9730
w/o Budgeted Selection	8.6500	1.1890
SSPEC, all three retained	**8.6100**	**0.9320**

Table 4: Failure modes per carrier: rate ↓.

Failure mode	Ask-before-Plan	ProClare	Minecraft
Mislabelled effect class	0.0633	0.0859	0.2112
Coincidental support span	0.0389	0.0849	0.1175
Deferred commit refused by budget	0.0324	0.0827	0.0760
Undeclared derivation reach	0.0302	0.0761	0.0704
Surrogate without an exact commit	0.0292	0.0500	0.0471

all three retained. The exposure split alone halves the exposure rate, and budgeted selection saves 0.2570 questions per episode. The split-off row confirms that Challenge #2 is answered by where the decision is taken and not by how the allowance is spent.

Failure Modes of an Unsettled Dispatch. Table 4 rates five failure modes on each carrier. The mislabelled effect class has the highest rate and the surrogate without an exact commit the lowest on all three carriers, at 0.0859 and 0.0500 on ProClare. The ordering locates the risk: a mislabelled effect defeats the dispatch rule of Eq. 8 at the point it is applied, whereas a surrogate with no exact commit is refused before the dispatch. On the Minecraft Dialogue Corpus the two leading modes reach 0.2112 and 0.1175, the highest rates in the table, while every mode on Ask-before-Plan stays below 0.07. The coincidental support span, a failure of the witness test of Eq. 3, ranks second on every carrier, at 0.0389, 0.0849 and 0.1175. Details in Appendix A.14.

6 Related Work

Deciding between acting and asking. Clarification policies read the act-or-ask choice from the instruction and the dialogue: by expected value (Rao & Daumé III, 2018), at the turn level (Alianejadi et al., 2019; Shi et al., 2022), over planning episodes (Zhang et al., 2024; Qian et al., 2024), or at every turn (Ye et al., 2025). Later work rewards questions that overturn a defeasible judgement (Pyatkin et al., 2023), and trains executors to ask when a handed instruction reads as unclear (Wang et al., 2025a; Zhang et al., 2025b). All read the unsettled set from an account of what the call still lacks; an interface-supplied value is a candidate only when that account includes it. SSPEC enumerates such values from the declaration and decides at the consuming dispatch.

Declared interfaces and argument construction. Tool declarations are studied as the abstraction calls are built against: collections become retrievable catalogues (Qin et al., 2024), decoding is constrained (Zhang et al., 2023), and editors revise the call's context (Li et al., 2025). Catalogues are also searched by self-reflective hierarchies (Du et al., 2024), and embedded reasoning raises parameter accuracy (Wei et al., 2026). Validity is the reported quantity: a call that passes counts as settled even when a default filled a field. SSPEC reads them as the enumeration surface, and a default-filled parameter stays a candidate until dispatch.

Control points and human oversight. Where a loop places its control point, and where a person sits, is a further axis: rubrics score reasoning steps (Kang et al., 2026), adaptation revises the action about to be taken (Yu et al., 2026), containment leaves holes for a supervisor (Zhao et al., 2026), and deployments place a person at correction time (He et al., 2025). Retrieved cases condition how far a loop reasons before acting (Pang et al., 2026), and others place a person at commissioning (Liu et al., 2026c) or at workflow repair (Wang et al., 2025b). All land on a stage boundary, none re-reading the value at the consuming action. SSPEC splits that trajectory at dispatch.

7 Conclusion

We present SSPEC, a framework that reads the ask-or-proceed decision off the declared interface: a recovery operator returns the fields no witness carries, an exposure split sorts them at the unrevertable dispatch, and budgeted selection spends the interruption allowance one unit per group. It reads 92.60 API match rate on default-filled-only episodes against 83.70 for the strongest published predictor and asks 1.6657 questions per ProClare episode against 3.1455 for that predictor. The declared interface carries the ask-or-proceed decision to values no executor report reaches.

REFERENCES

Misam Abbas. Attribution quality in ai-generated content: Benchmarking style embeddings and llm judges. In *ICDM (Workshops)*, 2025. doi: 10.1109/icdmw69685.2025.00205.

Seyed Morteza Ahmadian, Paolo Monti, and Carlos Natalino. A t-api-compliant react agentic loop for optical networks: Generic vs. domain-specific tool abstractions, 2026.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019. doi: 10.1145/3331184.3331265.

Lluis Castrejon, Thomas Mensink, Howard Zhou, Vittorio Ferrari, Andre Araujo, and Jasper Uijlings. Hammr: Hierarchical multimodal react agents for generic vqa, 2024.

Wei-Hao Chen, Weixi Tong, Amanda Case, and Tianyi Zhang. Dango: A mixed-initiative data wrangling system using large language model. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025. doi: 10.1145/3706598.3714135.

Yu Du, Fangyun Wei, and Hongyang Zhang. Anytool: Self-reflective, hierarchical agents for large-scale api calls. In *ICML*, 2024.

Yufei He, Ruoyu Li, Alex Chen, Yue Liu, Yulin Chen, Yuan Sui, Cheng Chen, Yi Zhu, Luca Luo, Frank Yang, and Bryan Hooi. Enabling self-improving agents to learn at test time with human-in-the-loop guidance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2025. doi: 10.18653/v1/2025.emnlp-industry.115.

Faramarz Jabbarvaziri. Remember, don't re-read: Stateful react agents for token-efficient autonomous experimentation, 2026.

Polaris Jhandi, Owais Kazi, Shreyas Subramanian, and Neel Sendas. Small language models for efficient agentic tool calling: Outperforming large models with targeted fine-tuning, 2025.

Jiazheng Kang, Bowen Zhang, Zixin Song, Jiangwang Chen, Xiao Yang, Da Zhu, and Guanjun Jiang. Co-ReAct: Rubrics as step-level collaborators for ReAct agents. arXiv preprint arXiv:2605.23590, 2026.

Henger Li, Shuangjie You, Flavio Di Palo, Yiyue Qian, and Ayush Jain. Verification-guided context optimization for tool calling via hierarchical llms-as-editors, 2025. URL `https://arxiv.org/abs/2512.13860`.

Aofan Liu, Haoxuan Li, Bin Wang, Ao Yang, and Hui Li. Ra-gen: A controllable code generation framework using react for multi-agent task execution. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2025. doi: 10.1109/ijcnn64981.2025.11228706.

Boyan Liu, Gongming Zhao, and Hongli Xu. Utility-guided agent orchestration for efficient llm tool use, 2026a.

Marianne Menglin Liu, Daniel Garcia, Fjona Parllaku, Vikas Upadhyay, Syed Fahad Allam Shah, and Dan Roth. ToolScope: Enhancing LLM agent tool use through tool merging and context-aware filtering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34095–34119, San Diego, California, United States, 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1573. URL `https://aclanthology.org/2026.acl-long.1573/`.

Zhihao Liu, Victor Nan Fernandez-Ayala, Tianyu Wang, Qiang Qin, Xi Vincent Wang, Dimos V. Dimarogonas, and Lihui Wang. Agentic neuro-symbolic planning and commissioning for human-in-the-loop industrial robotics with digital twins, 2026c.

Brielen Madureira and David Schlangen. Instruction clarification requests in multimodal collaborative dialogue games: Tasks, and an analysis of the codraw dataset. In *EACL*, 2023. doi: 10.18653/v1/2023.eacl-main.169.

Ali Montazeralghaem, Guy Tennenholtz, Craig Boutilier, and Ofer Meshi. Asking clarifying questions for preference elicitation with large language models, 2025.

Itay Nakash, George Kour, Guy Uziel, and Ateret Anaby-Tavor. Breaking react agents: Foot-in-the-door attack will get you in. In *NAACL (Findings)*, 2025.

Renning Pang, Tian Lan, Leyuan Liu, Piao Tong, Sheng Cao, and Xiaosong Zhang. Case-based calibration of adaptive reasoning and execution for llm tool use. In *ICCBR*, 2026.

Iman Peivaste and Salim Belouettar. From phase prediction to phase design: A react agent framework for high-entropy alloy discovery, 2026.

Valentina Pyatkin, Jena D. Hwang, Vivek Srikumar, Ximing Lu, Liwei Jiang, Yejin Choi, and Chandra Bhagavatula. ClarifyDelphi: Reinforced clarification questions with defeasibility rewards for social and moral situations. In *Annual Meeting of the Association for Computational Linguistics*, 2023. doi: 10.18653/v1/2023.acl-long.630.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.61.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*, 2024.

Ram Ramrakhya, Matthew Chang, Xavier Puig, Ruta Desai, Zsolt Kira, and Roozbeh Mottaghi. Grounding multimodal llms to embodied agents that ask for help with reinforcement learning, 2025.

Sudha Rao and Hal Daumé III. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2737–2746, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1255. URL https://aclanthology.org/P18-1255/.

Mohammadreza Rashidi. Depth-dependent indirect prompt injection in tool-calling react agents: Injection depth, payload framing, and turn-budget sensitivity, 2026.

Zhengxiang Shi, Yue Feng, and Aldo Lipani. Learning to execute actions or ask clarification questions. In *NAACL-HLT (Findings)*, 2022.

Hao Wang, Wenhui Zhu, Shao Tang, Zhipeng Wang, Xuanzhao Dong, Xin R. Li, Xiwen Chen, Ashish Bastola, Xinhao Huang, Yalin Wang, and Abolfazl Razi. Ezblender: Efficient 3d editing with plan-and-react agent. In *WACVW*, 2026.

Wenxuan Wang, Juluan Shi, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R. Lyu. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025a. doi: 10.18653/v1/2025.emnlp-main.1104.

Yafang Wang, Yangjie Tian, Xiaoyu Shen, Gaoyang Zhang, Jiaze Sun, He Zhang, Ruohua Xu, and Feng Zhao. Fault2flow: An alphaevolve-optimized human-in-the-loop multi-agent system for fault-to-workflow automation, 2025b.

Lei Wei, Jinpeng Ou, Xiao Peng, and Bin Wang. Think-augmented function calling: Improving llm parameter accuracy through embedded reasoning, 2026.

Jiaqi Wu, Qinlao Zhao, Zefeng Chen, Kai Qin, Yifei Zhao, Xueqian Wang, and Yuhang Yao. Gap: Graph-based agent planning with parallel tool use and reinforcement learning, 2025.

Jingheng Ye, Yong Jiang, Xiaobin Wang, Yinghui Li, Yangning Li, Pengjun Xie, and Fei Huang. ProductAgent: Benchmarking conversational product search agent with asking clarification questions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2025. doi: 10.18653/v1/2025.emnlp-industry.25.

Sheldon Yu, Junda Wu, Xintong Li, Nikki Lijing Kuang, Sizhe Zhou, Tong Yu, Jiawei Han, Jingbo Shang, and Julian J. McAuley. OLIVIA: Online learning via inference-time action adaptation for decision making in LLM ReAct agents. arXiv preprint arXiv:2605.11169, 2026.

Azizul Zahid, Subrata Biswas, Bashima Islam, and Sai Swaminathan. Procagent: An agentic framework for procedural task guidance on edge with human-in-the-loop, 2026.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Wang. Don't fine-tune, decode: Syntax error-free tool use via constrained decoding. arXiv preprint arXiv:2310.07075, 2023.

Michael J. Q. Zhang, W. Bradley Knox, and Eunsol Choi. Modeling future conversation turns to teach llms to ask clarifying questions. In *ICLR*, 2025a.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. Ask-before-plan: Proactive language agents for real-world planning. In *EMNLP (Findings)*, 2024.

Xuan Zhang, Yongliang Shen, Zhe Zheng, Linjuan Wu, Wenqi Zhang, Yuchen Yan, Qiuying Peng, Jun Wang, and Weiming Lu. AskToAct: Enhancing LLMs tool use via self-correcting clarification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025b. doi: 10.18653/v1/2025.emnlp-main.682.

Yaoyu Zhao, Yichen Xu, Oliver Bračevac, Cao Nguyen Pham, F. Wu, and Martin Odersky. Lacuna: Safe agents as recursive program holes, 2026.

Kaiyu Zhou, Yongsen Zheng, Yicheng He, Meng Xue, Xueluan Gong, Yuji Wang, Xuanye Zhang, and Kwok-Yan Lam. Beyond max tokens: Stealthy resource amplification via tool calling chains in llm agents, 2026. URL `https://arxiv.org/abs/2601.10955`.

AI USE STATEMENT

We used generative AI tools to translate the text (Claude Opus 5.0), to check grammar and spelling and polish the wording (GPT-5.6), to guide the drawing of the figures (Gemini), and to check the content (Grok). We did not use generative AI tools to modify code. We have reviewed all AI-assisted work: we checked every reference for hallucinated citations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The details needed to reproduce the results of this paper, including the datasets and episode regimes, the baseline instruments, the evaluation metrics, the hyperparameters and the full SSPEC procedure, are disclosed in this appendix.

APPENDIX A

Table 6 lists the contents of each part of the appendix.

APPENDIX A.0: LIMITATIONS

Interface detail. The paper lacks an evaluation on interfaces that state less about their arguments. Tool schemas range from defaults with units and frames down to argument names and types alone, or to free-form arguments, and the recovery operator draws its candidates from what the schema states. A reader who deploys the framework on a thinner interface needs to know how often it must flag an episode it can no longer serve there, and we have not carried out this evaluation.

Table 6: What each part of the appendix holds.

Section	Contents
Appendix A.0	Limitations
Appendix A.1	Related work
Appendix A.2	Motivation probes, in full resolution
Appendix A.3	Datasets and grids
Appendix A.4	Experimental tables and figures in detail
Appendix A.5	The simplest alternative and what it costs
Appendix A.6	Greedy selection and the minimum cover
Appendix A.7	The SSPEC procedure
Appendix A.8	Datasets and episode regimes
Appendix A.9	Baseline instruments in detail
Appendix A.10	Evaluation metrics, formally
Appendix A.11	Hyperparameters
Appendix A.12	Discussion
Appendix A.13	Future work
Appendix A.14	Further analysis of the body readings

Graphs with alternative derivations. The paper lacks a measurement of budgeted selection on dependency graphs in which a field can be derived from more than one parameter. The greedy cover is guaranteed to reach the minimum number of questions only when no field has such a second derivation. A reader who sets an interruption allowance needs to know how far the greedy cover stands above that minimum on these graphs, and we have not measured it.

User perception of the questions. The paper lacks a user study of how people experience the questions the framework asks. Every result is read from annotated episodes, and a reader who places the framework in front of users needs to know how those users judge the timing and the wording of each question, and we have not run such a study.

Appendix A.1: Related Work

Deciding between acting and asking. Clarification policies read the choice between acting and questioning from the instruction and the dialogue so far. Early formulations propose to rank candidate questions by their expected value to the task (Rao & Daumé III, 2018), introduce a reward for questions that overturn a defeasible judgement (Pyatkin et al., 2023), or train a turn-level executor that learns to act or ask on collaborative building dialogue (Shi et al., 2022; Madureira & Schlangen, 2023). Agentic successors keep that carrier and widen the setting, with a predictor learned over annotated planning episodes (Zhang et al., 2024), executors that learn to ask when a handed instruction reads as unclear (Wang et al., 2025a; Zhang et al., 2025b), and a simulator of future turns developed to supply the training signal (Zhang et al., 2025a). Interactive systems apply the same decision in other settings: it is applied to product search (Ye et al., 2025), studied for preference elicitation (Montazeralghaem et al., 2025), applied to data wrangling (Chen et al., 2025), and examined for embodied help-seeking (Ramrakhya et al., 2025). Each reads the unsettled set from an account of what the call still lacks. SSPEC enumerates the values in force from the declared interface instead, and decides at the dispatch that consumes each one.

Declared interfaces and argument construction. Tool declarations are themselves the object of a second line, which asks which abstraction of them yields calls an agentic loop can emit and an interface will accept. Large API collections are organised into retrievable catalogues (Qin et al., 2024), searched by self-reflective hierarchies (Du et al., 2024), merged and pruned before the model sees them (Liu et al., 2026b), and ordered by an estimate of what each call is worth (Liu et al., 2026a). Comparisons of generic against domain-specific abstractions report which declaration style a loop can satisfy, in an operations setting and under a hierarchical controller designed to route to the sub-agent holding the matching interface (Ahmadian et al., 2026; Castrejon et al., 2024). Other work targets the argument itself: constrained decoding removes syntax errors (Zhang et al., 2023), embedded reasoning is shown to raise parameter accuracy (Wei et al., 2026), proposed editor hierarchies revise the context a call is built from (Li et al., 2025), and compact models learn to emit

the same calls (Jhandi et al., 2025). Validity is the reported quantity. A call that passes is recorded as settled even where the declared default, and no decision, supplied one of its fields. SSPEC reads those declarations as the enumeration surface for values in force, and an interface-filled parameter stays a candidate until the call goes out.

Control points inside agentic loops. Where a loop places its control point forms a further axis. Step-level rubrics score intermediate reasoning (Kang et al., 2026), inference-time adaptation revises the action about to be taken (Yu et al., 2026), and retrieved cases are shown to condition how far the loop reasons before it executes (Pang et al., 2026). Graph-structured planners are introduced to expose parallel branches for scheduling (Wu et al., 2025), while stateful loops carry earlier findings forward rather than re-reading them (Jabbarvaziri, 2026). Domain deployments developed for code generation (Liu et al., 2025), scene editing (Wang et al., 2026), and alloy design (Peivaste & Belouettar, 2026) instantiate the same skeleton. All land on the selecting step. None re-reads a value at the later action that consumes it: a field fixed early travels unchallenged to its use. SSPEC splits that trajectory at the dispatch boundary and settles a dependency where its consumption cannot be undone.

Unrevertable effects and human oversight. Safety work asks what a loop may do with no person in it. Attack studies trace how an injected instruction reaches a tool call and how deep into a chain it survives (Nakash et al., 2025; Rashidi, 2026), and how chained calls can be driven to consume resources (Zhou et al., 2026). Containment proposals leave holes in the program for a supervising process to fill (Zhao et al., 2026). Deployments that keep a person in the loop are designed to place that person at commissioning (Liu et al., 2026c), at procedural guidance (Zahid et al., 2026), at test-time correction (He et al., 2025), and at workflow repair (Wang et al., 2025b). Separately, attribution benchmarks compare embedding scorers against model judges on whether generated content rests on a cited source (Abbas, 2025). Oversight lands at a stage boundary. Nobody is asked about one argument of the imminent call, and the instruments that do ask question the executor that wrote it. SSPEC accepts a value only when a witness reproduces it from a cited span and a type-preserving substitution on that span changes the answer, a test that consults the evidence and the schema and never the producer.

APPENDIX A.2: MOTIVATION PROBES

The three panels of Figure 2 are probes. Each one is read on its own subset, and no probe row enters any table of the main results. This section gives each panel at full resolution, the numbers behind it, and what the panel is read for.

The left panel: the carrier ceiling. Figure 5 plots, for the three instruments scored on the travel-planning rows, the clarification accuracy the instrument reports on its own decisions against the part of the annotated unsettled set those decisions touch. Table 7 lists the two columns. The missing-argument trigger reports 22.6000 accuracy at 0.0565 coverage, the ask-or-execute policy 41.0000 at 0.3458, and executor self-report 76.8000 at 0.3238. Accuracy rises monotonically across the three while coverage does not, and the widest coverage in the panel, 0.3458, belongs to the middle instrument rather than to the most accurate one. The two axes are therefore not two readings of one quality: the first records how often an instrument is right about whether to ask, the second how much of what needed asking it ever ranged over. The call arguments carry the same asymmetry, since the part of call arguments the executor did not write itself stands at 0.8963 on the tool-use episodes of that carrier.

Table 7: Reported accuracy against coverage of the annotated unsettled set, on travel planning.

Instrument	Reported clarification accuracy ↑	Coverage of the unsettled set ↑
Missing-argument trigger	22.6000	0.0565
Ask-or-execute (Shi et al., 2022)	41.0000	0.3458
Executor self-report	76.8000	0.3238

The middle panel: the deferred cost. Figure 6 plots, for each published case, the number of executed steps between the step that fixes a value and the first consumption of it that cannot be

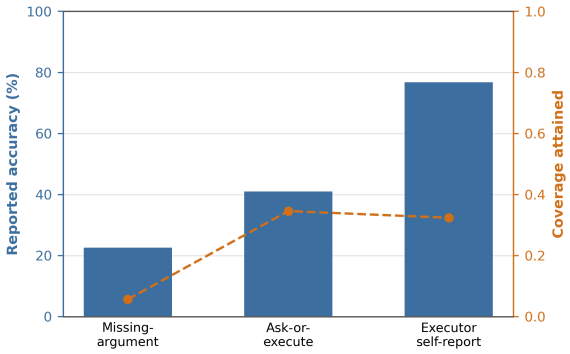

Figure 5: Figure 2(left) at full resolution.

undone, beside the number of dependencies still unsettled when that consumption arrives. Table 8 lists both columns. The distance is 1.67 steps on a product order dispatch, 3.57 on a block placement rollback, 2.62 on a tool retry after failure and 1.89 on a budget-constrained booking: the interval in which a question is still cheap exists in every case. The right axis does not track the left one: the rollback case carries the longest distance and the fewest unsettled dependencies, 0.1596, while the retry and booking cases carry more than one unsettled dependency each at shorter distances. A policy that judged exposure at the writing step would have to predict both quantities before the consuming action is chosen.

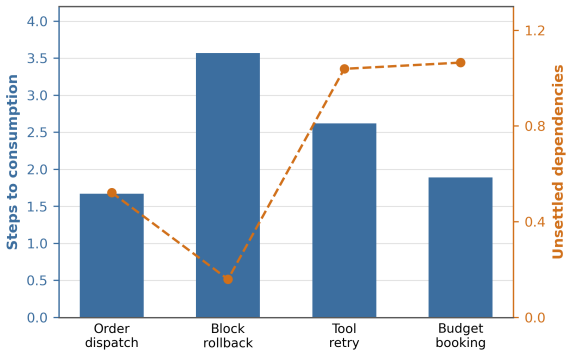

Figure 6: Figure 2(middle) at full resolution.

Table 8: Distance from fixing a value to its first unrevertable use, per case, in executed steps.

Published case	Steps to first unrevertable use	Unsettled dependencies there
Product order dispatch	1.67	0.5222
Block placement rollback	3.57	0.1596
Tool retry after failure	2.62	1.0392
Budget-constrained booking	1.89	1.0653

The right panel: the coupled questions. Figure 7 plots, for each carrier, the count of separately underspecified fields per episode beside the number of fields that one declared unit or frame parameter closes at once. Table 9 lists both columns. Travel planning carries 0.5517 fields per episode against a reach of 3.21, product search 0.6000 against 20.00, and building dialogue 0.0760 against 12.83. The reach exceeds the count by a factor of 5.8 on travel planning, 33 on product search and 169 on building dialogue, which is the quantity a field-by-field ranking pays for repeatedly.

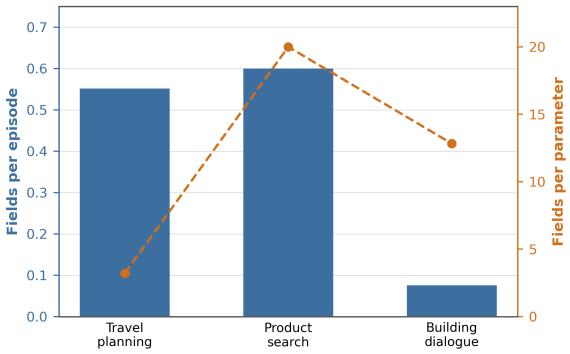

Figure 7: Figure 2(right) at full resolution.

Table 9: Question load and coupling per carrier.

Published carrier	Underspecified fields per episode	Fields reached by one parameter
Travel planning (Zhang et al., 2024)	0.5517	3.21
Product search (Ye et al., 2025)	0.6000	20.00
Building dialogue (Shi et al., 2022)	0.0760	12.83

Appendix A.3: Datasets and Grids

Dataset sizes and splits follow the statistics table of each source. The body grids include Table 1, Table 2 and Table 5, with Figure 4 plotting clarification accuracy and API match rate across regimes.

Appendix A.4: Experimental Tables and Figures in Detail

This section gives, for the grids and plot printed in the body, their reading conventions.

Table 1. The rows are the five compared policies of Section 5 and SSPEC, and the columns are API match rate and final pass rate, both of them a percentage on which higher is better. The per-column best is set in bold and the last row is this work. The episodes are the default-filled-only subset of the Ask-before-Plan carrier.

Figure 4. The left panel plots clarification accuracy over the first three specification regimes and the right panel API match rate over all four, both in percent, for the policies of Table 1 other than the no-ask executor. The four regimes are those named in Appendix A.8: fully specified, one unsupported value, several unsupported values, and default filled only. The bold red line is SSPEC. The right-hand column of the right panel is the regime Table 1 reports, and the two floats agree cell by cell on API match rate.

Table 2. Task completion is the percentage of episodes ProClare scores as completed and questions per episode is a count, with lower better on the question column and higher better on completion. The irreversible exposure rate is a percentage on which lower is better. The per-column best is in bold and the last row is SSPEC. The no-ask executor asks nothing by construction, which is why it holds the question column.

The ablation switches. The switches of the ablation grid are read against the components of the method as follows: without the exposure split judges exposure once, at the step that fixes the value, without the reversible surrogate verifies every unrevertable call with a question, and without budgeted selection ranks single fields instead of groups.

Table 5. The rows switch the three online components on and off on the Minecraft Dialogue Corpus carrier. The pairwise row is read in Appendix A.12. The irreversible exposure rate is a

percentage and questions per episode is a count, and lower is better on both, and the bold marking follows the lower value. The last row is the full configuration.

Appendix A.5: The Simplest Alternative and What It Costs

Asking the executor which arguments it assumed is the executor self-report instrument. Two further alternatives bound it from either side. The missing-argument trigger asks only when a required argument has no value at all, which is the rule a schema validator already implements. The every-turn ask spends a question on every turn regardless of what the interface declares, following the clarifying agent of Ye et al. (2025). Three episode regimes separate the cases these instruments are meant to tell apart: episodes in which every value is supported, episodes carrying one value fixed without support, and episodes whose only unsupported values are the ones a declared default has filled in silently.

Table 3 reads both quantities for all four instruments. Recall of self-report is 48.5% once a single unsupported value is present and falls to 38.6% when the unsupported values are default-filled, while the framework reads 82.1% and 68.4% in the same two regimes. Self-report degrades where the executor has less to report on, since a default-filled argument looks answered from inside the executor.

Cost separates the two instruments that reach high recall. The every-turn ask attains 99.4% recall in both regimes, the highest figure in the table, and pays 5.0000 interruptions per episode for it. The framework spends 0.0302, 0.3054 and 0.3138 interruptions per episode across the three regimes, more than an order of magnitude below the every-turn rule. Self-report spends 0.6359 to 0.6499 interruptions per episode and returns less than the framework does in both regimes with reported recall: it is dominated on both quantities rather than trading one against the other. A reader weighing the simpler alternative therefore pays for it twice, once in values never recovered and once in questions spent on episodes that needed none.

Appendix A.6: Greedy Selection and the Minimum Cover

This section proves the statement on budgeted selection for one forced set H and gives an instance in which the greedy cover exceeds the minimum. The derivation graph has the unit and frame parameters of the interface and the fields of the call as nodes, with an edge from a node to every node derived directly from it. The group of a parameter p is the set of fields reachable from p, which is what one question about p settles, and a field that no parameter reaches forms a group on its own; $\mathcal{G}(H)$ holds every group that meets H. A node admits an alternative derivation when it has more than one incoming edge. Settling H follows Eq. 7: while the set H' of fields of H still open is non-empty, the next question asks about a group $g^\star$ maximising $|g^\star \cap H'|$, and the fields of $g^\star \cap H'$ then leave H'.

No alternative derivation. When no node admits an alternative derivation, the greedy rule settles H with the minimum number of questions that any sequence of groups needs, whichever maximiser it picks. Every node then has at most one incoming edge, so the nodes from which a field is reachable lie on the single backward walk of incoming edges that ends at that field. If the groups of two parameters share a field, both parameters lie on that walk, one of them reaches the other, and its group contains the other group, while the group of a field that no parameter reaches meets no other group; any two groups are therefore nested or disjoint. Let $M_1, \ldots, M_k$ be the members of $\mathcal{G}(H)$ that are maximal under inclusion. They are pairwise disjoint, every field of H lies in one of them, and every member of $\mathcal{G}(H)$ lies inside exactly one of them. A group that contains a field of M_i lies inside M_i, so a sequence that settles H asks at least one question inside each M_i and needs at least k questions. When the greedy rule picks $g^\star \subseteq M_i$, the group M_i is itself a candidate and $g^\star \cap H' \subseteq M_i \cap H'$, so the maximality of $g^\star$ forces $g^\star \cap H' = M_i \cap H'$: the question closes every open field of M_i and none outside it. Each question therefore empties one M_i that still holds open fields, and the greedy rule stops after exactly k questions.

With alternative derivations. Let parameter x derive fields f_1, f_2, f_3, parameter y derive f_4, f_5, f_6, and parameter z derive f_2, f_3, f_4, f_5, so that each of f_2 to f_5 admits an alternative derivation, and let $H = \{f_1, \ldots, f_6\}$. The groups of x and y settle H with two questions. The greedy rule

Algorithm 1 Dispatch-Time Ask-or-Proceed Decision (SSPEC)

Input: interface declaring, for each action a, its schema $\Sigma(a)$, defaults δ_a, consumed fields $\mathrm{dep}(a)$, effect class $\mathrm{irrev}(a)$, reversible alternatives $\mathcal{A}_r(a)$ and step cost $\mathrm{cost}(a)$; witness family $\mathcal{W}$; group enumerator $\mathcal{G}$; allowance b
Output: the delivered artefact, with every field of L stated beside it
Initialize $b_t \leftarrow b$, $L \leftarrow \emptyset$
while the episode has an imminent action a_t **do**
    Read the state s_t, its evidence buffer B_t, and the emitted fields E_t with assignment x_t
    // Recovery operator R, Section 4.2
    $\Phi(s_t) \leftarrow \mathrm{dom}\,\Sigma(a_t)$, $C_t \leftarrow \Phi(s_t) \setminus E_t$; read v_t over $\Phi(s_t)$ via Eq. 2
    for each $f \in \Phi(s_t)$ **do**
        Compute the support indicator $\widehat{G}_t(f)$ over $\mathcal{W}(B_t)$ via Eq. 3
    end for
    $\widehat{S}_t \leftarrow R\big(\Phi(s_t), B_t\big) = \{\, f \in \Phi(s_t) : \widehat{G}_t(f) = 0 \,\}$ via Eq. 4
    // Dispatch-time exposure, Section 4.3
    $H_t \leftarrow \{\, f \in \widehat{S}_t : f \in \mathrm{dep}(a_t),\ \mathrm{irrev}(a_t) \,\}$, $L_t \leftarrow \widehat{S}_t \setminus H_t$ via Eq. 5; $L \leftarrow (L \setminus \Phi(s_t)) \cup L_t$
    // Dispatch rule Δ of Eq. 8, branches read in order
    if $H_t = \emptyset$ **then**
        Issue a_t
    else if $b_t \geq 1$ **then**
        Select $g^\star$ over $\mathcal{G}(H_t)$ via Eq. 7; ask $g^\star$ and commit its answer to the evidence the next step reads; $b_t \leftarrow b_t - 1$
    else
        if no $a \in \mathcal{A}_r(a_t)$ has $\mathrm{dep}(a) \supseteq H_t$ **then**
            Leave a_t unissued and end the episode
        else
            Select $\tilde{a}_t$ via Eq. 6 and run it
            if its outcome fixes every $f \in H_t$ to one value $\Sigma(a_t)$ admits **then**
                Issue a_t with those values (exact commit)
            else
                Leave a_t unissued and end the episode
            end if
        end if
    end if
end while
return the artefact with every field of L stated beside it

first asks about z, which closes four fields against three for x or for y; no group then holds both f_1 and f_6, and two further questions follow, three in total against a minimum of two.

Appendix A.7: The SSPEC Procedure

This section fixes the order in which the three components run over one imminent call and the point at which the allowance is charged. For each call the field domain the declared schema puts in force is enumerated and every field of it is paired with the value in force, whether the call emitted that value or the interface supplied its declared default; each field is then put to the discriminating witness test, and the fields no witness carries are retained. The retained set is split at the same dispatch into the part the call consumes under an unrevertable effect class and the remainder, a split recomputed at every dispatch rather than fixed when a value was written. The dispatch rule is then read in the three branches of Eq. 8 in order, and the allowance is charged in one place only, on the branch where the hard-exposed part is non-empty and a unit remains.

Appendix A.8: Datasets and Episode Regimes

This section states the carriers behind the datasets paragraph of Section 5 and the partition the regime plots are read over. The four regimes group the episodes of a carrier by how the arguments of the imminent call are settled: fully specified, one unsupported value, several unsupported values, and default-filled only. The last of these is the regime in which every unsettled argument is supplied by a declared default, and it is the regime Table 1 reports.

APPENDIX A.9: BASELINE INSTRUMENTS IN DETAIL

This section states what each compared policy does with the call it faces. The clarification predictor of Zhang et al. (2024) decides whether to ask from the dialogue state before the call is composed. The ask-or-execute policy of Shi et al. (2022) decides at the turn level between asking and acting. The every-turn-ask policy of Ye et al. (2025) puts a question at every turn regardless of what the episode has settled. The executor self-report instrument asks the model which arguments it assumed and treats the reported ones as unsupported, which bounds the comparison from the side of the executor's own account. The no-ask executor issues every call as composed and bounds it from the side of asking nothing.

APPENDIX A.10: EVALUATION METRICS, FORMALLY

This section gives the definitions behind the metrics paragraph of Section 5. The first three metrics follow the tool-learning and clarification evaluation of Zhang et al. (2024); the remaining three are introduced by this work and are defined over the same episodes. Let E be the evaluated episodes, C the calls issued over them, d_e the ask-or-proceed decision taken in episode e against the annotated decision $d_e^\star$, and x_c the argument assignment of call c against the reference assignment $x_c^\star$.

$$\text{Clarification accuracy} = \frac{1}{|E|}\sum_{e\in E}\mathbf{1}[d_e = d_e^\star]\,, \tag{9}$$

$$\text{API match rate} = \frac{1}{|C|}\sum_{c\in C}\mathbf{1}[x_c = x_c^\star]\,, \tag{10}$$

$$\text{Final pass rate} = \frac{1}{|E|}\sum_{e\in E}\mathbf{1}[e \text{ passes}]\,, \tag{11}$$

$$\text{Unsupported-value recall} = \frac{\sum_t |\widehat{S}_t \cap S_t|}{\sum_t |S_t|}, \tag{12}$$

where S_t is the set of Eq. 1 and $\widehat{S}_t$ is the set the scored instrument flags at step t, which for SSPEC is what the recovery operator returns. The remaining two are read at the dispatch the call is issued from,

$$\text{Interruptions per episode} = \frac{1}{|E|}\sum_{e\in E} Q_e(\pi), \tag{13}$$

$$\text{Irreversible exposure rate} = \frac{\sum_t \mathbf{1}[\mathrm{irrev}(a_t) \wedge \mathrm{dep}(a_t) \cap S_t \neq \emptyset]}{\sum_t \mathbf{1}[\mathrm{irrev}(a_t)]}. \tag{14}$$

Lower is better on the irreversible exposure rate and the interruptions per episode, and higher is better on every other metric; $Q_e(\pi)$ is the number of questions π asks in episode e.

APPENDIX A.11: HYPERPARAMETERS

This section states the settings behind the hyperparameters paragraph of Section 5. The witness family $\mathcal{W}(B_t)$ enumerates the contiguous spans of the evidence buffer that the episode has already committed: a candidate witness is a span of user text or of a returned tool result rather than a generated paraphrase. The substitution that tests discrimination replaces such a span with an alternative of the same declared type drawn from the schema of the field under test, which keeps the call valid and isolates the span as the reason the value is what it is. The effect class $\mathrm{irrev}(a_t)$ is taken from the carrier's own declaration of the action rather than inferred from its name, and the episodes whose declaration is wrong are reported as a separate failure mode rather than repaired. Equal-cost covers in budgeted selection are resolved by one fixed tie-breaking rule applied identically to every SSPEC arm. The interruption allowance b is held at one value across every SSPEC arm of every table.

APPENDIX A.12: DISCUSSION

Exposure and question cost answer to different parts of the framework, and the online ablation separates them cleanly. Holding the reversible surrogate and the budgeted selection in place while

removing the exposure split of Eq. 5 moves the exposure rate from 8.6100 to 17.2500 while the question count barely moves, 0.9320 against 0.9935. Keeping the split and removing the two cost-side components reverses the pattern: interruptions per episode rise to 2.2300 at an exposure rate of 8.8000. The two quantities a deployment trades off are therefore tunable one at a time: a carrier whose actions are rarely unrevertable can keep the split and drop the surrogate search.

The framework rests on a property of the tools rather than of the executor. The compared policies do not read defaults, units and frames off the declared schema at the moment the ask-or-proceed decision is taken (Zhang et al., 2024; Shi et al., 2022). Separating the enumeration stage from the allowance stage is what lets the two costs move apart. A question can only be spent on an argument the recovery operator has already returned; what the interface supports reading bounds what any allowance policy can protect afterwards. Overall, the takeaway is that what a tool interface already declares is a more readable source for the ask-or-proceed decision than what the executor can report about its own assumptions.

Appendix A.13: Future Work

A next step is to replace the exact-reproduction acceptance rule with a typed rule that admits the substitutions a schema declares equivalent and to score the recovered set under it on unsupported-value recall at a fixed interruption allowance.

Several wider directions follow. Replacing the greedy cover with an exact minimum cover on graphs in which a node admits an alternative derivation would remove the extra questions the greedy rule can ask there. Carriers whose actions are densely unrevertable, such as file-system and code-execution agents, would test generality where the exposure split has more to absorb. A deployment study that measures corrective turns after delivery would give direct evidence that stating an undetermined value alongside the outcome reduces downstream repair.

Appendix A.14: Further Analysis of the Body Readings

Failure modes carrier by carrier. On ProClare the mislabelled effect class at 0.0859 stands above the coincidental support span at 0.0849, the deferred commit refused by budget at 0.0827, the undeclared derivation reach at 0.0761 and the surrogate that reaches no exact commit at 0.0500. The building task holds the same two ends, 0.2112 against 0.0471, and on travel planning the coincidental support span reads 0.0389 while the undeclared derivation reach reads 0.0302. On travel planning the mislabelled effect class reads 0.0633. The surrogate that reaches no exact commit is the rarest mode on all three carriers.

The ceiling of questioning the executor. Executor self-report, which asks the executor what it assumed, is scored here on the same episodes. It recovers 38.6% of the unsupported set on default-filled episodes against 68.4% for the framework. The published instruments sit on the same two axes: an instrument can report high clarification accuracy on its decisions to ask or proceed while reaching a small share of the annotated unsettled set, which is the reading the motivation panel rests on. The ceiling is a property of the instrument's range and not of its accuracy. More details about the simplest alternative and its question cost are provided in Appendix A.5.

Cost of one dispatch. Table 10 breaks the cost of resolving one call into the stages of the framework; it is set by those stages and by nothing the executor reports. The acceptance test of Eq. 3 dominates: a candidate witness is a committed span of the evidence buffer, the family grows quadratically in the length of that buffer, and each field is tested against it twice, once for reproduction and once for the substitution. The remaining stages are passes over sets the call and its interface already declare, and the dispatch rule is constant. Restricting the witness family to spans the episode has already committed, rather than to arbitrary paraphrases, is what keeps the dominant term tractable.

Table 10: Cost of each stage of the framework; n is the length of the evidence buffer B_t.

Stage	Eq.	Cost
Interface complement	2	linear in $\lvert\Phi(s_t)\rvert$
Discriminating witness	3	two tests per field over $O(n^2)$ spans
One-sided closure	4	linear in $\lvert\Phi(s_t)\rvert$
Exposure split	5	linear in $\lvert\widehat{S}_t\rvert$
Budgeted selection	7	one pass over $\mathcal{G}(H_t)$ per question
Reversible surrogate	6	one pass over $\mathcal{A}_r(a_t)$
Dispatch rule	8	constant