



WikiPlanner: wiki-derived temporal monitors for safe LLM agent planning - a medical-domain proof of concept

Bailing Zhang

Keywords:

LLM agent planning, temporal logic, runtime verification, knowledge-based constraints, clinical decision support

Citation:

Zhang, B. WikiPlanner: wiki-derived temporal monitors for safe LLM agent planning - a medical-domain proof of concept. *AI Agent* 2026, 2, 25. <https://dx.doi.org/10.20517/aiagent.2026.22>

Received: 7 Jun 2026

First Decision: 23 Jul 2026

Revised: 30 Jul 2026

Accepted: 16 Sep 2026

Published: 14 Oct 2026

Academic Editor:

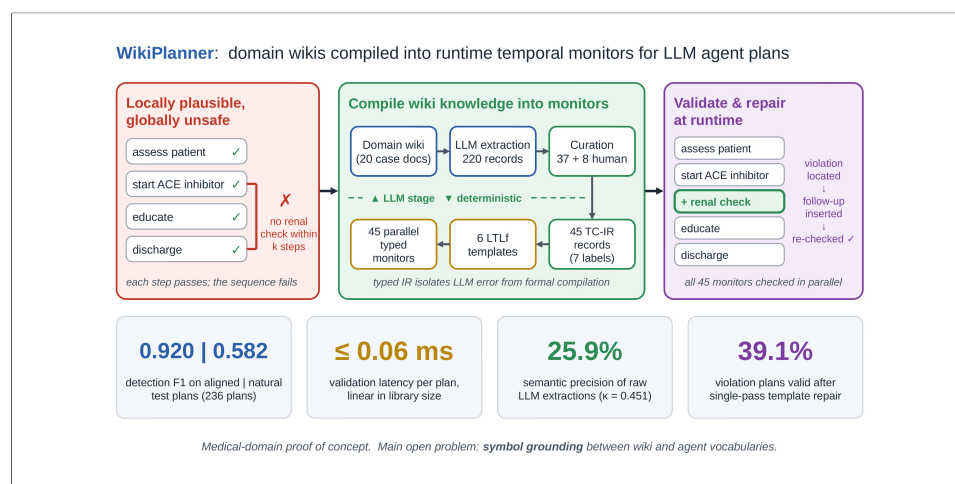
Aloysius Soon

Copy Editor:

Tong Wang

Production Editor:

Tong Wang



Abstract

Large language model (LLM) agents in high-stakes sequential planning may produce locally reasonable steps whose overall sequence violates domain constraints; single-step filters cannot detect such cross-step temporal violations. We present WikiPlanner, which extracts temporal constraints from domain wikis, structures them via a typed Temporal Constraint Intermediate Representation (TC-IR) with seven constraint labels compiled to six Linear Temporal Logic over finite traces (LTLf) templates, and instantiates them as parallel runtime monitors. The two-stage pipeline - TC-IR extraction followed by deterministic template compilation - isolates LLM extraction uncertainty from formal compilation. We evaluate every stage on a medical planning benchmark of 45 constraints and 236 test plans (137 aligned, 99 natural). A dual-annotator audit of the 220 raw extractions ($\kappa = 0.451$, adjudicated) places semantic extraction precision at 25.9%; an automated check finds 79.5% of these records structurally well-formed (90.1% on a held-out nursing corpus); invented orderings, state-vs.-action confusion, and silent parse loss from output truncation are the dominant error classes. Plan validation achieves F1 = 0.920 (95%CI: 0.873-0.956) on the aligned set and F1 = 0.582 (95%CI: 0.447-0.701) on the natural set; the gap is attributed primarily to constraint-library coverage and symbol grounding, as wiki-derived constraint symbols and generator vocabulary form largely



Institute of Data Science and Intelligent Technology, NingboTech University, Ningbo 315336, Zhejiang, China.

Correspondence to: Dr. Bailing Zhang, Institute of Data Science and Intelligent Technology, NingboTech University, Ningbo 315336, Zhejiang, China. E-mail: bailing.zhang@nit.zju.edu.cn

disjoint namespaces. Latency scales linearly with library size (0.006-0.059 ms). Single-pass template repair resolves 39.1% of violation plans (mean 1.30 word-level edits per success); local post-trigger repairs substantially outperform pre-trigger modifications. Results are a single-domain proof of concept, and symbol grounding is the main obstacle to natural-plan deployment.

INTRODUCTION

Deploying large language model (LLM) agents in consequential domains requires more than generating locally plausible actions: the complete action sequence must conform to domain-specific temporal ordering rules. A clinician agent that prescribes an angiotensin-converting enzyme (ACE) inhibitor may act on sound local reasoning, but failure to schedule renal function monitoring within the following visits violates a standard safety guideline that independent per-step checking cannot detect. Similar issues arise in industrial control, pharmaceutical management, and regulated service workflows, where the order and timing of actions carry safety implications that no single step expresses in isolation.

Recent work on steering LLM outputs has focused on constitution-guided training and feedback^[1], prompt-based self-verification^[2], and retrieval grounding of generation^[3]. Runtime verification in classical software engineering^[4,5] checks execution traces against formal specifications, but has not been systematically adapted to settings where constraints derive from natural-language domain wikis and must be applied to LLM-generated action sequences. Closer to our problem, Agent-C^[6] enforces temporal safety properties during LLM token generation using satisfiability modulo theories (SMT) solving, achieving perfect conformance on retail and airline reservation tasks; RvLLM^[7] performs runtime verification of LLM outputs against domain-specific predicates encoded in a custom specification language. Both works assume that constraints are manually authored by domain experts or encoded as task specifications.

WikiPlanner addresses a complementary knowledge-engineering problem: how to automatically derive temporal constraints from maintained domain wikis and compile them into reusable runtime monitors, without requiring manual specification authoring. Domain wikis encode rich temporal ordering knowledge - prerequisite relationships, mandatory follow-up requirements, time-bounded monitoring windows, mutual exclusions, and forbidden action sequences - that can serve as machine-enforceable constraints if compiled into an appropriate formal representation.

The core technical challenge is that direct LLM-to-formal-logic compilation is unreliable: LLMs confuse temporal direction, misspecify time bounds, and produce invalid formula syntax. WikiPlanner addresses this by introducing a Temporal Constraint Intermediate Representation (TC-IR), a typed JavaScript Object Notation (JSON) schema that captures constraint semantics before any formal compilation. LLMs target TC-IR extraction, and a deterministic template compiler then maps each record to a Linear Temporal Logic over finite traces (LTLf) formula implemented as a parallel typed monitor.

This study is deliberately scoped as a single-domain proof of concept: all empirical claims are established on one medical planning benchmark, and we make no generality claims beyond it. Within that scope, however, we evaluate every stage of the pipeline, including the wiki-to-constraint extraction stage that is the hardest to measure and the most consequential for what the monitors can enforce.

This paper makes the following contributions:

- We formulate the problem of wiki-derived temporal runtime monitoring for LLM agent plans, where constraints are automatically extracted from maintained domain wiki sources rather than manually specified, and instantiate it in a medical planning domain.

- We introduce TC-IR, a typed intermediate representation that isolates LLM extraction uncertainty from deterministic monitor compilation, comprising seven constraint labels compiled to six LTLf templates (RESPONSE and MANDATORY_FOLLOWUP share a template), together with a complete formal definition and well-formedness conditions.
- We provide a two-part evaluation of extraction quality: a dual-annotator precision audit of all 220 raw extractions with inter-annotator agreement and an error taxonomy, and a transfer study applying the identical pipeline to a held-out, publicly released corpus to test structural generalization and label-distribution transfer under domain shift - addressing the pipeline stage that is hardest to measure and most often left unevaluated.
- We implement a family of parallel typed finite-trace monitors enabling source-localized violation reporting without product-automaton construction.
- We provide both controlled and natural-plan evaluations, with a per-case decomposition of the aligned-to-natural performance gap that separates constraint-library coverage, symbol grounding, ground-truth label noise, and monitor logic as distinct causes.
- We analyze single-pass template repair and show that repairs acting locally after the trigger succeed far more often than repairs modifying pre-trigger context, with failures dominated by multi-violation residuals and constraint-coupling cascades.

RELATED WORK

LLM planning and its limitations

Multiple studies document systematic failures of LLMs in classical planning. Hybrid pipelines such as LLM+P^[8] respond by delegating plan synthesis to classical planners; Kambhampati *et al.* argue that LLMs are best used as heuristic generators within modular planning frameworks^[9]; Valmeekam *et al.* show that LLMs frequently violate global constraints while generating locally plausible steps^[2]. LLM hallucinations^[10] compound this problem in knowledge-intensive domains.

Runtime monitoring and temporal logic

Runtime verification^[4,5] checks execution traces against formal specifications. Classical Linear Temporal Logic (LTL)^[11] targets infinite traces; De Giacomo and Vardi^[12] introduced LTLf for finite traces, admitting deterministic finite automaton (DFA) representations via tools such as ltlf2dfa^[13]. Enforcing temporal constraints on generative agents using Temporal Stream Logic (TSL) has been explored by Rothkopf *et al.*^[14], achieving over 96% adherence vs. 14.67% for unconstrained LLMs on structured agent tasks. WikiPlanner adopts LTLf as its formal backbone but replaces general DFA compilation with typed monitors implementing the acceptance condition per constraint class, avoiding product-automaton state explosion.

Temporal constraint enforcement for LLM agents

Agent-C^[6] presents a framework that provides runtime guarantees for LLM agents by translating manually authored temporal properties into first-order logic and using SMT solving to enforce compliance during token generation, achieving 100% conformance on retail and airline service tasks. RvLLM^[7] verifies LLM outputs against domain-specific constraints encoded in Expert Specification Language (ESL), a custom specification language, targeting single-output violations in legal and numerical domains. Both require expert-authored constraint specifications.

WikiPlanner is complementary: rather than enforcing manually specified constraints, it addresses the knowledge engineering problem of automatically compiling constraints from domain wikis into reusable monitors. This distinction - wiki-sourced, automatically extracted, sequence-level - defines WikiPlanner's position relative to both Agent-C and RvLLM.

Knowledge bases and wiki-as-constraint

Retrieval-augmented generation (RAG)^[3] augments LLMs with retrieved text at inference time, but retrieved passages are not enforced as hard constraints. Karpathy argues that structured wiki layers maintained separately from the generative model provide verifiable, updatable knowledge for agent systems^[15]. WikiPlanner operationalizes this view by treating wiki content as a compilable constraint specification rather than a retrieval source.

THE WIKIPLANNER FRAMEWORK

Problem setting

Let A denote a finite action vocabulary and let a plan $\pi = (a_1, a_2, \dots, a_n)$ be an ordered sequence of actions $a_i \in A$. A domain wiki W is a collection of natural-language documents encoding temporal ordering knowledge about actions in A . WikiPlanner aims to: (i) detect whether π violates any constraint derivable from W ; and (ii) repair violations by producing a minimally modified plan π' satisfying all hard constraints.

Definition 1 (TC-IR record). A TC-IR record is a tuple $c = (\tau, \alpha, B, k, \sigma, \rho)$, where $\tau \in T$ is one of seven constraint labels, $T = \{\text{PREREQUISITE}, \text{MANDATORY_FOLLOWUP}, \text{BOUNDED_FOLLOWUP}, \text{MUTUAL_EXCLUSION}, \text{FORBIDDEN_SEQUENCE}, \text{ORDERING}, \text{RESPONSE}\}$; $\alpha \in A$ is the trigger action; B is a type-dependent map of related-action roles to elements of A (prior_action for PREREQUISITE; followup_action for MANDATORY_FOLLOWUP, BOUNDED_FOLLOWUP, and RESPONSE; excluded_action for MUTUAL_EXCLUSION; forbidden_action and optional_guard_action for FORBIDDEN_SEQUENCE; must_come_after for ORDERING); $k \in \mathbb{N} \cup \{\perp\}$ is a step bound, required if and only if $\tau = \text{BOUNDED_FOLLOWUP}$; $\sigma \in \{\text{hard}, \text{soft}\}$ is the severity; and $\rho \in [0, 1]$ is the extractor-assigned confidence. A record is well-formed if and only if (i) all referenced actions are non-empty snake_case identifiers; (ii) the role keys in B match τ as specified above; (iii) k is a positive integer when $\tau = \text{BOUNDED_FOLLOWUP}$ and $\perp$ otherwise; and (iv) $\rho \in [0, 1]$. The schema validator rejects records violating any of (i)-(iv).

An important scope boundary: WikiPlanner processes ordinal temporal constraints (action ordering and within- k -step requirements) rather than real-time metric constraints. Time-bounded requirements (e.g., “within 48 h”) are discretized to step counts under a one-step-per-clinical-contact assumption, which is appropriate for clinic-scale planning but would require extension to Metric Temporal Logic (MTL) for settings with variable inter-step durations.

Framework overview

Figure 1 shows the four-stage pipeline, with stages numbered S1-S4 and visually grouped. (S1 Extract) GLM-4-flash reads chunked wiki text and outputs 220 raw TC-IR records, of which 211 pass schema validation. (S2 Curate and Compile) Post-processing (compile-error repair and targeted re-extraction) yields a 239-record pool, confidence-ranked curation selects 37 records, and 8 manually authored, human-verified records are added to form the 45-record library; a deterministic template compiler then maps each library record to an LTLf formula. (S3 Instantiate) Each formula is instantiated as an independent typed monitor. (S4 Validate and Repair) Incoming LLM plans are grounded to vocabulary symbols and validated against all monitors in parallel; violations trigger the repair module.

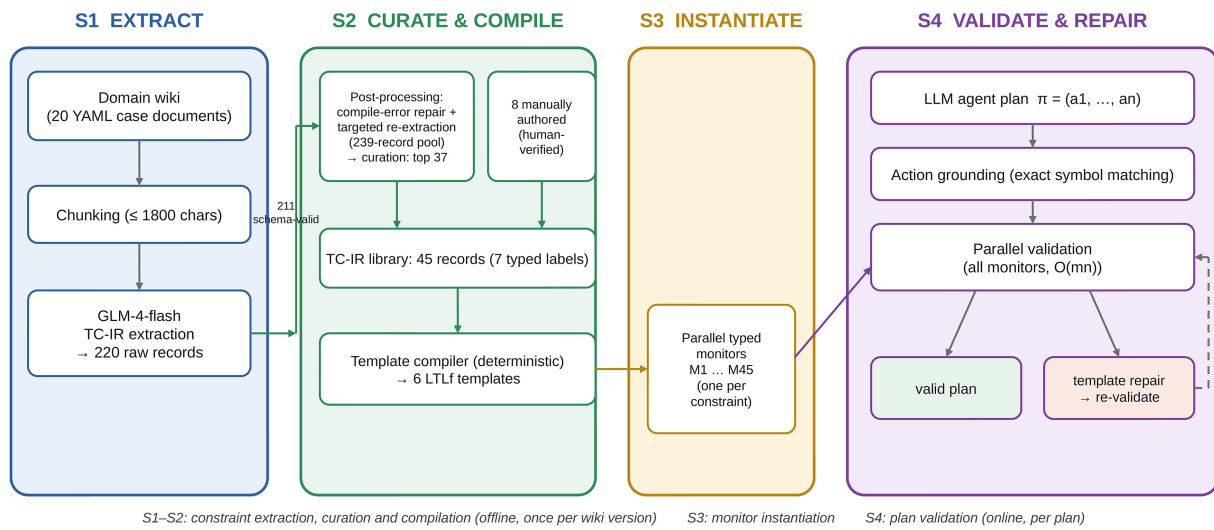


Figure 1. WikiPlanner four-stage pipeline. Functional blocks are grouped and numbered by stage: S1 Extract (wiki text $\rightarrow$ 220 raw TC-IR records via GLM-4-flash; 211 schema-valid), S2 Curate and Compile (post-processing and targeted re-extraction $\rightarrow$ 239-record pool; confidence-ranked curation $\rightarrow$ 37 records; + 8 manually authored, human-verified records $\rightarrow$ 45-record library; deterministic template compiler $\rightarrow$ LTLf formulas), S3 Instantiate (parallel typed monitors), S4 Validate and Repair (symbol grounding, parallel validation, repair module). LLM: Large language model; TC-IR: Temporal Constraint Intermediate Representation; LTLf: Linear Temporal Logic over finite traces.

Terminology. The following conventions are used throughout. A constraint label is one of the seven values of τ in Definition 1 (PREREQUISITE through RESPONSE). A TC-IR record is a single extracted tuple; the constraint library is the curated set of 45 records deployed in all experiments. An LTLf template is one of the six parameterized formula schemas of the compiler; applying a template to a record yields that record's instantiated formula φ_i , and the typed monitor M_i is the executable checker implementing φ_i . The plan validator is the parallel composition of all instantiated monitors (distinct from the schema validator of Definition 1, which checks record well-formedness). An action symbol is an element of the action alphabet A ; the action vocabulary is the 40-symbol subset of A available to the plan generator.

TC-IR schema and constraint classes

The TC-IR is a typed JSON record separating LLM-based extraction from deterministic LTLf compilation, formalized in Definition 1. Table 1 defines the seven TC-IR labels and the six LTLf templates they compile to. RESPONSE is consolidated into the MANDATORY_FOLLOWUP template at the formula level [both express $G(\alpha \rightarrow F\beta)$], while the label distinction is retained in TC-IR for clinical annotation: RESPONSE marks emergency or safety-triggered requirements (e.g., anaphylaxis $\rightarrow$ epinephrine) while MANDATORY_FOLLOWUP marks protocol-level follow-up [e.g., angiotensin-converting enzyme inhibitor (ACEI) $\rightarrow$ estimated glomerular filtration rate (eGFR) check]. Both enforce the same temporal logic but differ in severity and clinical urgency, which downstream triage systems may exploit. We therefore write “seven TC-IR labels, six LTLf templates” throughout.

A design note on MUTUAL_EXCLUSION: under single-action-per-step semantics, the superficially natural template $G(\neg(\alpha \wedge \beta))$ is vacuously satisfied, since α and β can never hold at the same step. The template is therefore $\neg(F\alpha \wedge F\beta)$, which rejects any trace containing both α and β at any positions.

Action grounding

LLM-generated plans and TC-IR records both denote actions by snake_case identifiers, but the two symbol spaces need not coincide. In the current implementation, plan steps are grounded to constraint symbols by exact string matching: a monitor consults a step only if the step string equals the constraint's trigger or

Table 1. TC-IR constraint labels, operational semantics, and LTLf templates

Label	Operational semantics	LTLf formula
PREREQUISITE	Reject at α if β has not been observed before	$\neg \alpha \text{ W } \beta$
MANDATORY_FOLLOWUP	Reject at end of trace if no β occurs after α	$G(\alpha \rightarrow F\beta)$
BOUNDED_FOLLOWUP	Reject if β does not appear within k steps after α	$G(\alpha \rightarrow \bigvee_{i=0}^k X^i \beta)$
MUTUAL_EXCLUSION	Reject if both α and β appear anywhere in trace	$\neg(F\alpha \wedge F\beta)$
FORBIDDEN_SEQUENCE	After α , reject if β appears before guard γ	$G(\alpha \rightarrow \neg(\neg\gamma \text{ U } \beta))$
ORDERING	Reject at β if α has not occurred	$\neg\beta \text{ W } \alpha$
RESPONSE*	Emergency/safety-triggered follow-up; same template as MANDATORY_FOLLOWUP	$G(\alpha \rightarrow F\beta)$

α = trigger, β = related action, k = step bound, γ = guard action. Seven labels compile to six LTLf templates; RESPONSE and MANDATORY_FOLLOWUP share $G(\alpha \rightarrow F\beta)$ and differ in TC-IR severity and clinical semantics. TC-IR: Temporal Constraint Intermediate Representation; LTLf: Linear Temporal Logic over finite traces.

related action identifier, and steps outside the monitored symbol set are simply never matched. No semantic normalization layer is applied. This design choice makes grounding behavior fully transparent and turns the symbol-space overlap between plans and constraints into a directly measurable quantity, which we report in the Results: on average 66.0% of aligned-set steps and 74.5% of natural-set steps fall outside the monitored symbol set, and the disjointness of the two namespaces - rather than synonym-level near-misses alone - is the dominant cause of the aligned-to-natural performance gap.

Parallel typed monitors

Given a TC-IR record set $C = \{c_1, \dots, c_m\}$, each record instantiates one typed monitor. All monitors run in parallel over the plan; the plan validator collects every violation rather than stopping at the first, enabling source-localized violation reporting.

Definition 2 (Typed monitor). A typed monitor M_i for constraint c_i is a function $M_i: A^* \rightarrow \{0,1\} \times \mathbb{Z}_{\geq -1}$ that directly implements the acceptance condition for the LTLf formula ϕ_i associated with c_i . In the codomain, $\mathbb{Z}_{\geq -1}$ denotes the set of integers ≥ -1 : the value -1 is a sentinel for “no violation”; non-negative values are the zero-based index of the first violating step.

Each monitor runs in $O(n)$ time over the plan (n = plan length). Total validation cost is $O(mn)$ where $m = |C|$. The sum of DFA state counts $\sum |Q_i|$ across all monitors is substantially smaller than the product $\prod |Q_i|$ that a single merged DFA would require, avoiding state-space explosion. One degenerate parameterization deserves note: a FORBIDDEN_SEQUENCE record whose trigger equals its forbidden action (a “self-loop” constraint) is not monitorable by the evaluated implementation, because re-occurrence of the trigger re-arms rather than violates the monitor; the two aligned-set misses of this type are analyzed in the failure-mode section, and a correction of the edge case is supplied as a patch in the released repository.

Two-stage repair

Upon detecting hard violations, WikiPlanner attempts two-stage repair. Stage 1 (template repair) applies a type-specific single edit targeting the first detected violation: PREREQUISITE violations insert the missing prior action before the trigger (insert_prerequisite); MANDATORY_FOLLOWUP and RESPONSE insert the missing follow-up after the trigger (insert_followup); BOUNDED_FOLLOWUP moves the follow-up action into the required window (move_to_window); MUTUAL_EXCLUSION removes the excluded action (remove_excluded); FORBIDDEN_SEQUENCE inserts the guard action between trigger and forbidden action (insert_guard) or removes the forbidden action (remove_forbidden); ORDERING transposes the two actions (reorder). Every template edit is anchored at the position of the triggering action of the first detected

violation; when that action does not occur in the plan at all - possible for ORDERING, whose monitor also fires when the dependent action appears without its trigger - no anchor exists, no edit is applied, and the case is logged as `no_template_matched`. The repaired plan is re-validated immediately, and the repair is committed only if all hard violations are cleared. Note that because the repair targets the first detected violation, the applied method may correspond to a constraint whose label differs from the plan's labeled violation type; both views are reported in the Results. Stage 2 (A* search) is included as a preliminary fallback for residual violations, bounded to five single-step edits over the domain vocabulary; it is evaluated exploratorily rather than as a primary contribution.

MATERIALS AND METHODS

Wiki corpus and preprocessing

The source corpus comprises 20 de-identified YAML (YAML Ain't Markup Language) case documents from two educational medical wikis: a general practice (GP) clinic wiki (10 files) and a medical school wiki (10 files), covering chronic disease management, psychiatry, emergency medicine, pharmacology, and clinical examination skills. Each YAML document is flattened to plain text and split into chunks of at most 1,800 characters at newline boundaries, yielding 85 chunks in total. No other preprocessing is applied. The complete file list and per-document chunk counts are given in the [Supplementary Materials](#).

TC-IR extraction

Each chunk is submitted to GLM-4-flash - a lightweight production variant of the GLM-4 model family^[16], served through the ZhipuAI open platform (<https://open.bigmodel.cn>) - with a fixed system prompt (reproduced verbatim in the [Supplementary Materials](#)) instructing the model to return a JSON array of TC-IR records, with temperature 0.1, max_tokens 1,024, up to three retries with exponential back-off on transport errors, and a 1-second inter-call interval. Responses are stripped of Markdown fences and parsed as JSON; responses that fail to parse contribute no records. This raw pass produced 220 candidate records. A post-processing stage then (i) repairs systematic schema deviations (e.g., ORDERING records carrying a `guard_action` key are reclassified as FORBIDDEN_SEQUENCE; PREREQUISITE records carrying `must_come_after` are re-keyed); (ii) performs targeted re-extraction for under-represented labels; and (iii) deduplicates on the (trigger_action, constraint_type) key, yielding a fixed and enriched pool of 239 records. From this pool, a per-label curation step selects the highest-confidence records up to per-label targets (PREREQUISITE 10, ORDERING 10, MANDATORY_FOLLOWUP 8, BOUNDED_FOLLOWUP 6, MUTUAL_EXCLUSION 5, FORBIDDEN_SEQUENCE 5, RESPONSE 5), producing 37 curated constraints; the BOUNDED_FOLLOWUP, MUTUAL_EXCLUSION, and RESPONSE pools were exhausted below target (1, 1, and 2 available, respectively), which quantifies the label-scarcity phenomenon that motivated manual supplementation. A confidence threshold of 0.60 is implemented in the pipeline, but every extracted record carried confidence at or above 0.8, so the threshold is inactive on this corpus and the effective selection mechanism is per-label confidence-ranked curation. Eight additional constraints covering the under-represented BOUNDED_FOLLOWUP, MUTUAL_EXCLUSION, and RESPONSE labels were manually authored and human-verified, giving the final 45-constraint library. Zero LTLf compilation errors were observed on the final library.

Action vocabulary

Plan generation uses a fixed 40-symbol action vocabulary, released in full with the paper. The vocabulary was constructed by normalizing the clinically executable actions recurring across the wiki corpus into snake_case identifiers spanning history-taking, examination, investigation, prescribing, monitoring, and referral, and is listed in full in the [Supplementary Materials](#). The 45 constraints reference 61 distinct action symbols, of which only 15 also occur in the 40-symbol generation vocabulary; the remaining 46 are wiki-idiosyncratic identifiers (e.g., `red_flag`, `urgent_referral`, `hypertensive_emergency`). This deliberate separation between the two symbol spaces is measured, rather than hidden, by the grounding statistics reported in the Results.

Test plan generation and labeling

Two test sets were generated with GLM-4-flash under a strict-vocabulary protocol: generated steps are checked against the 40-symbol vocabulary and out-of-vocabulary generations trigger a constrained retry. The aligned set (137 plans: 50 correct, 87 violations) uses constraint-specific violation directives referencing the target constraint's own symbols (e.g., "include prescribe_acei but not check_egfr"), ensuring ground-truth alignment with monitored constraints; because directives inject constraint symbols verbatim, aligned plans may contain identifiers outside the 40-symbol vocabulary. The natural set (99 plans: 50 correct, 49 violations) generates plans from clinical scenario prompts without constraint-specific directives, modeling deployment where the generator does not know which action pairs are monitored. Violation labels and free-text violation descriptions are assigned by the generator at creation time from its own clinical knowledge; they are not restricted to the 45-constraint library. This labeling choice makes the natural set an open-world test of the deployed library and, as the Results show, introduces a measurable amount of label noise that we report rather than filter.

Baselines

Seven validators are compared, varying only in constraint source and checking mode with all other settings fixed: B0 (no validation; accepts every plan), B1 (per-action danger list: rejects plans containing any of a fixed list of dangerous solo actions), B2 (bag-of-words action-presence rules), B3 (10 handcrafted sequential rules authored directly in the plan vocabulary), B4 (LLM self-check: GLM-4-flash is shown the plan and asked to judge validity; temperature 0.05, same model as extraction), B5 (WikiPlanner-Lite: the 45 TC-IR constraints with action-presence checking only), and WikiPlanner-Full (all 45 constraints with sequential typed monitors); WikiPlanner-Verified uses only the 8 human-verified constraints. Metrics: precision, recall, F1, false rejection rate ($FRR = FP/(FP+TN)$), and per-plan latency. Latency is measured on an Intel Core i7-9750H @ 2.60 GHz laptop (Python 3.12) as the mean of 10 repeated passes over the aligned set; F1 confidence intervals are 95% bootstrap intervals over 1,000 plan-level resamples.

Repair metrics

Repair is evaluated on the 87 aligned violation plans. A repair attempt is successful if re-validation reports no hard violations. Edit cost is reported as the word-level Levenshtein distance between the original and repaired sequence (insertion/deletion cost 1; a move therefore costs 2). Levenshtein distance is used in preference to net length change, which assigns zero cost to successful move-type repairs and can therefore yield means below one edit per applied repair. Failure causes are logged per case as cascade (the repair introduces at least one new violation) or residual (original violations remain, typically in multi-violation plans under single-pass repair).

EXTRACTION QUALITY EVALUATION

The extraction stage is the only component of the pipeline whose output is not deterministic, and it is therefore the component whose quality most directly bounds what the monitors can enforce. We evaluate it in two complementary parts. Part 1 audits the precision of the raw extraction output on the source corpus; Part 2 measures precision and recall against human gold annotation on a held-out corpus, avoiding the circularity of building and evaluating the library on the same documents.

Part 1: dual-annotator precision audit of 220 raw extractions

All 220 raw records (before any post-processing) were independently judged by two annotators - a senior clinician-researcher and a medical student - on four dimensions: Q1, whether the record constitutes a genuine temporal rule between two clinical actions as stated by its own natural-language field (requiring

both that the source sentence itself asserts the temporal relation, and that the structured fields are faithful to it with executable actions as participants); Q2, label correctness; Q3, temporal direction correctness; and Q4, bound plausibility (applicable to BOUNDED_FOLLOWUP only; no raw record carried this label, so Q4 is vacuous in this audit). Annotation followed a written guideline; an initial calibration round revealed a systematic divergence in the reading of Q1 (“clinically plausible ordering” vs. “ordering asserted by the source sentence”), after which the guideline was clarified with worked examples and the full set was re-annotated independently. Nine guideline exemplar items are excluded from agreement statistics. On the remaining 211 items, raw agreement on Q1 was 76.8% with Cohen’s $\kappa = 0.451$ (moderate); conditional on both annotators judging an item valid, agreement was 91.7% on Q2 and 97.2% on Q3. The 49 Q1 disagreements - 42 of which follow the pattern senior = valid/student = invalid, concentrated on “implied ordering” records extracted from examination-sequence and rationale text - were resolved by third-party adjudication against the written criteria: 19 were adjudicated valid and 30 invalid, with the adjudicator upholding the senior annotator on 23 items, the student on 21, and reaching an independent judgment differing from both on 5.

Adjudicated extraction precision is $57/220 = 25.9\%$. The numerator comprises the 48 of the 211 audited items that were valid after adjudication (29 judged valid by both annotators plus 19 of the 49 adjudicated disagreements) together with the nine guideline exemplar items, which are valid by construction and excluded only from the agreement statistics; excluding the exemplars gives $48/211 = 22.7\%$. Roughly one in four raw extractions is thus a well-formed temporal rule faithful to its source sentence. An independent structural check - applied programmatically and requiring no human judgment - finds 79.5% of the same 220 records structurally well-formed (having a second action, type-consistent role keys, a non-degenerate action pair, and a bound where the label requires one), confirming that the dominant failure is semantic rather than syntactic: records can satisfy the schema while asserting an ordering the source text never states. Among the 133 records both annotators independently judged invalid, the error taxonomy separates T1 errors - the source sentence asserts no temporal relation, so the extractor invents an ordering from enumerative, procedural, or explanatory text (66 records) - from T2 errors - the structured fields are unfaithful to the sentence, or a participant is a state, diagnosis, cognitive activity, or skill rather than an executable action (42 records), with 25 records failing both tests. T1 dominance is the single most actionable finding for extraction-prompt design. The audit also explains why the deployed library retains only 37 of 239 post-processed records: per-label confidence-ranked curation implicitly filters much of the invalid mass, and this audit makes that filtering measurable rather than assumed.

Part 2: held-out structural generalization study

To test whether extraction behavior transfers beyond the source wikis, we applied the identical pipeline (same model, prompt, chunking, and parameters) to a held-out corpus of 10 de-identified perioperative nursing documents from a publicly released repository, covering antithrombotic management, antibiotic prophylaxis timing, instrument and sponge counting, sterile technique, surgical time-out, post-anesthesia care, hypothermia management, wound care, venous thromboembolism prophylaxis, and documentation. The corpus is disjoint from the source wikis in specialty, document style, and provenance. Extraction produced 144 chunks and 252 parsed records, of which 227 (90.1%) are structurally well-formed under the same programmatic criteria - higher than the 79.5% observed on the source corpus, indicating that structural quality does not degrade under domain shift. The label distribution is again dominated by ORDERING (133) and PREREQUISITE (99), with MANDATORY_FOLLOWUP (9), FORBIDDEN_SEQUENCE (8), and BOUNDED_FOLLOWUP (3) sparse and MUTUAL_EXCLUSION and RESPONSE absent entirely, independently reproducing the label-scarcity pattern that motivated manual supplementation of the deployed library. The held-out run also exposes a loss mode invisible in aggregate counts: 48 of 144 responses contained TC-IR records but failed JSON parsing because generation hit the max_tokens limit

mid-array, and a bracket-matching repair recovers 391 further records (643 total). Since truncation is biased toward rule-dense chunks, this silent parse loss is a distinct and consequential error class inherited unchanged from the source-corpus pipeline; we report headline figures for the pipeline as evaluated (252 records) and treat the recovery figure as diagnostic. We did not construct a human gold annotation for this corpus, so neither extraction precision nor recall against a human reference is measured on it: the held-out study assesses structural generalization and label-distribution transfer only, and this is stated as a limitation rather than estimated. The extraction outputs summarized here are available from the corresponding author on reasonable request.

RESULTS AND DISCUSSION

Main results

Table 2 reports results on the aligned test set with per-cell confusion counts. WikiPlanner-Full achieves $F1 = 0.920$ (95%CI: 0.873-0.956), improving by 0.268 over the best handcrafted sequential baseline (B3, $F1 = 0.652$) and by 0.224 over the best ordering-agnostic baseline (WikiPlanner-Lite, $F1 = 0.696$). Equal precision and recall (0.920) indicate calibrated coverage across violation types. LLM self-checking (B4) reaches $F1 = 0.524$ (precision 0.846, recall 0.379) at 3,055 ms mean latency - five orders of magnitude slower than monitor validation - and its per-type detection is weakest precisely on cross-step obligation types (FORBIDDEN_SEQUENCE 0/7, MANDATORY_FOLLOWUP 3/16, BOUNDED_FOLLOWUP 2/8), underscoring the complementarity of formal monitors. Table 3 breaks down WikiPlanner-Full detection by constraint label on the aligned set: the three fully detected labels (BOUNDED_FOLLOWUP, MUTUAL_EXCLUSION, RESPONSE 8/8 each) are exactly those anchored by human-verified constraints, while FORBIDDEN_SEQUENCE (5/7) is capped by the self-loop edge case analyzed below.

Natural-set results and gap decomposition

Table 4 reports the complete baseline comparison on the natural set. WikiPlanner-Full achieves $F1 = 0.582$ (precision 0.767, recall 0.469, FRR 0.140; 95%CI: 0.447-0.701). Two results demand explanation. First, the 0.338 gap to the aligned set. A per-case analysis of all 26 false negatives attributes 17 to constraint-library coverage gaps (the generator labeled violations of clinical rules - suicide-risk assessment before monoamine oxidase inhibitor (MAOI) initiation, glucose testing before diabetes medication, blood-pressure assessment before beta-blockade - for which no constraint exists in the 45-record library), 7 to ground-truth label noise (the violation description contradicts the step sequence, or an omission is tagged with an unrelated label), 2 to symbol grounding (a semantically corresponding constraint exists but the plan expresses the trigger concept under a different identifier), and 0 to monitor logic. Second, and more systematically: only 7 of 45 constraints ever have their trigger symbol appear in any natural plan, and 0 of the 37 auto-extracted constraints are activatable on the natural set at all; on average 74.5% of natural-plan steps (66.0% of aligned-plan steps) lie outside the monitored symbol set, and 46 of the 61 action symbols referenced by constraints do not occur in the 40-symbol generation vocabulary. The aligned set overcomes this disjointness only because violation directives inject constraint symbols into the plans. Synonym-level vocabulary mismatch, a natural first hypothesis for this gap, accounts for only 2 of the 26 cases; the decomposition shows the operative structure is namespace disjointness plus library coverage.

This analysis also explains the second notable result: the handcrafted baseline B3 outperforms WikiPlanner-Full on the natural set (0.818 vs. 0.582). B3's ten rules are written directly in the plan vocabulary and are therefore fully grounded by construction. Far from undermining the approach, this inversion isolates the bottleneck: once symbols are grounded (the aligned set), typed monitors over the wiki-derived library dominate every baseline including B3 (0.920 vs. 0.652); when they are not, no amount of temporal-logic machinery can fire. Symbol grounding between wiki-derived constraints and generator vocabularies - not temporal monitoring itself - is the deployment-critical problem, and co-designing the two symbol spaces (or

Table 2. Aligned test set (137 plans: 87 violations, 50 correct)

Method	TP	FP	TN	FN	Precision	Recall	F1	FRR	Latency (ms)
B0 No validation	0	0	50	87	—	0.000	—	0.000	0.0
B1 Single-step rules	12	2	48	75	0.857	0.138	0.238	0.040	0.0
B2 Bag-of-words	35	3	47	52	0.921	0.402	0.560	0.060	0.0
B3 Handcrafted seq. (10)	46	8	42	41	0.852	0.529	0.652	0.160	0.0
B4 LLM self-check	33	6	44	54	0.846	0.379	0.524	0.120	3055.5
B5 WikiPlanner-Lite	48	3	47	39	0.941	0.552	0.696	0.060	0.0
WikiPlanner-Verified (8)	43	7	43	44	0.860	0.494	0.628	0.140	< 0.1
WikiPlanner-Full (45)	80	7	43	7	0.920	0.920	0.920	0.140	0.059

Confusion counts are provided for transparency; all methods vary only in constraint source and checking mode. B0 precision and F1 are undefined (0/0) and shown as “—”. Latency for WikiPlanner rows: mean of 10 repeated passes, Intel Core i7-9750H. TP: True positives; FP: false positives; TN: true negatives; FN: false negatives; FRR: false rejection rate.

inserting a learned grounding layer, evaluated as rigorously as the rest of the pipeline) is the corresponding research agenda.

Ablation study

A1: Component ablation. Table 5 decomposes F1 contributions by constraint source (handcrafted vs. wiki-derived) and checking mode (bag-of-words vs. sequential) on the aligned set. Sequential checking adds +0.092 F1 over handcrafted rules and +0.224 F1 over wiki-derived rules; the interaction between wiki-derived constraints and sequential checking exceeds the additive prediction by +0.132 F1. This super-additive effect arises because wiki-derived constraints cover constraint labels (e.g., BOUNDED_FOLLOWUP) that are invisible to bag-of-words checking but fully exploited by sequential monitors.

A2: Constraint source. Auto-extracted constraints alone ($n = 37$) achieve $F1 = 0.766$ with $FRR = 0.000$; human-verified constraints alone ($n = 8$) yield $F1 = 0.628$ with $FRR = 0.140$; the combined library achieves $F1 = 0.920$ with $FRR = 0.140$. The revision adds the mechanistic reading: the zero false-rejection rate of auto-extracted constraints is not evidence of superior calibration but of limited activation - their wiki-idiosyncratic symbols rarely match plan steps except where violation directives inject them, so they rarely fire on correct plans either. All seven false rejections originate from the human-verified constraints, whose symbols are grounded in the plan vocabulary and therefore can fire pre-emptively (see failure modes).

A3: Scalability. Constraint subsets are formed by adding human-verified constraints first and then auto-extracted constraints in descending confidence order. F1 increases monotonically from 0.621 to 0.920 across subset sizes (5, 10, 15, 20, 30, 45) and per-plan latency scales linearly from 0.006 ms to 0.059 ms [Intel Core i7-9750H; mean $\pm$ standard deviation (SD) over 10 repetitions], with bootstrap 95% CIs shown as error bars in Figure 2.

Table 3. Per-type violation detection rates for WikiPlanner-Full on the aligned test set

Constraint type	Detected/Total	Detection rate
BOUNDED_FOLLOWUP	8/8	100%
MUTUAL_EXCLUSION	8/8	100%
RESPONSE	8/8	100%
PREREQUISITE	19/20	95%
ORDERING	18/20	90%
MANDATORY_FOLLOWUP	14/16	88%
FORBIDDEN_SEQUENCE	5/7	71%
Overall	80/87	92%

A4: Template repair. Table 6 reports repair outcomes on the 87 aligned violation plans with the corrected edit metric and per-case failure logging. Overall, 34/87 (39.1%) of plans validate after single-pass repair; however, 7 of these successes are vacuous (the plan validator did not detect the labeled violation, so no repair was applied), and the strict success rate over the 80 plans where repair was applicable is 27/80 (33.8%). The mean word-level Levenshtein cost of the 27 applied successful repairs is 1.30. Aggregating by repair method rather than violation type reveals the operative pattern: repairs that act locally after the trigger - insert_followup (9/16), move_to_window (12/22), remove_excluded (4/4), remove_forbidden (1/1) - succeed at 26/43, whereas repairs that modify pre-trigger context or introduce new symbols - insert_prerequisite (1/21), reorder (0/5), insert_guard (0/2) - succeed at 1/28. Classifying repairs by edit operation alone (insertion versus reordering) obscures this pattern, since insert_prerequisite is an insertion yet fails almost always; what separates success from failure is whether the edit disturbs the plan prefix preceding the trigger. Per-case logs show two failure mechanisms: residual violations in multi-violation plans under single-pass repair (39/53 failures) and constraint-coupling cascades (14/53), in which the inserted prior action - typically a wiki symbol such as red_flag - immediately activates its own MANDATORY_FOLLOWUP obligations. Figure 3 shows the distribution of applied repair methods: move_to_window (22) and insert_prerequisite (21) together account for 49.4% of all 87 violation plans (53.8% of the 80 applicable cases), and nine ORDERING cases are logged as no_template_matched because the triggering action of the first detected violation is absent from the plan, leaving the template edit no anchor (see Two-stage repair). The other seven ORDERING residual failures did receive an applied edit: five reorder repairs cleared the first detected ORDERING violation but left other already-detected violations standing, and two move_to_window repairs (the first detected violation being BOUNDED_FOLLOWUP) cleared that violation while the ORDERING violations remained; none of the sixteen introduced a new violation, so all sixteen are residual rather than cascade failures, consistent with the case-level repair log.

Failure mode analysis

Four classes of failure characterize WikiPlanner's current limitations.

(1) Symbol grounding failure. Wiki-derived constraint symbols and generator vocabulary form largely disjoint namespaces (46/61 constraint symbols absent from the generation vocabulary; 0/37 auto-extracted constraints activatable on natural plans). Grounding failures range from case-sensitivity (call_000_or_immediate_ed vs. ..._ED, one aligned miss) through synonym mismatch (perform_physical_exam vs. physical_examination) to abstract wiki symbols that generators never emit (red_flag, symptom). Five of the seven aligned-set misses and the dominant share of the natural-set gap are grounding failures. Resolving this requires co-design of the two symbol spaces or an explicitly evaluated grounding layer.

Table 4. Natural test set (99 plans: 49 violations, 50 correct), complete baseline comparison

Method	TP	FP	TN	FN	Precision	Recall	F1	FRR
B0 No validation	0	0	50	49	–	0.000	–	0.000
B1 Single-step rules	11	2	48	38	0.846	0.224	0.355	0.040
B2 Bag-of-words	20	3	47	29	0.870	0.408	0.556	0.060
B3 Handcrafted seq. (10)	36	3	47	13	0.923	0.735	0.818	0.060
B4 LLM self-check	15	6	44	34	0.714	0.306	0.429	0.120
B5 WikiPlanner-Lite	20	3	47	29	0.870	0.408	0.556	0.060
WikiPlanner-Full (45)	23	7	43	26	0.767	0.469	0.582	0.140

B0 precision and F1 are undefined (0/0) and shown as “–”. B3’s advantage stems from being authored directly in the plan vocabulary; see text. TP: True positives; FP: false positives; TN: true negatives; FN: false negatives; FRR: false rejection rate.

(2) Pre-emptive monitoring and schema insufficiency. Seven correct plans are falsely rejected because they perform a required check before prescribing (e.g., `check_egfr` → `prescribe_acei` → `schedule_followup`), while the `BOUNDED_FOLLOWUP` monitor requires the check after the trigger. The TC-IR schema cannot express bidirectional temporal windows (“before or after”). All false rejections originate from the eight human-verified constraints - precisely because they are the grounded ones (see A2).

(3) Temporal granularity mismatch. Discretizing clock-time windows to step counts is adequate under the one-step-per-contact assumption but inappropriate for variable inter-step durations; MTL extension would remove the assumption at higher cost.

(4) Constraint provenance and authority. The TC-IR schema does not capture provenance (source document, evidence span, version, verification status), limiting auditability of violation reports. Adding source metadata is a concrete near-term improvement.

One implementation edge case is reported separately from these design-level classes: the self-loop `FORBIDDEN_SEQUENCE` parameterization (trigger = forbidden action) is silently unmonitorable in the evaluated code because trigger re-occurrence re-arms the monitor before the forbidden-branch test; this accounts for both `FORBIDDEN_SEQUENCE` misses in Table 3 (5/7). A patch that fixes the branch order is included in the released repository; applying it changes the verdict of exactly these two plans and no other. All figures reported here reflect the implementation as evaluated.

Limitations

Six limitations deserve explicit statement. First, the aligned-set evaluation ($F1 = 0.920$) uses violation plans generated with constraint-specific directives; it validates monitor correctness but overstates deployment performance relative to the natural set ($F1 = 0.582$). Second, all experiments cover a single medical planning domain at

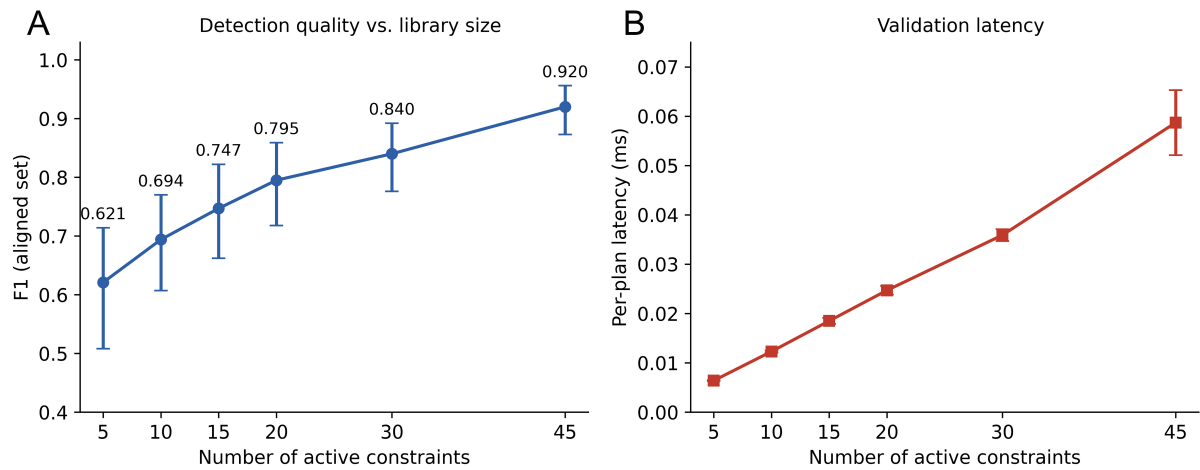


Figure 2. (A) Aligned-set F1 versus number of active monitors; error bars are bootstrap 95% CIs (1,000 plan-level resamples); (B) Per-plan validation latency versus number of active monitors; error bars are ± 1 SD over 10 repeated passes (Intel Core i7-9750H, Python 3.12). Subsets add human-verified constraints first, then auto-extracted constraints in descending confidence order. SD: Standard deviation.

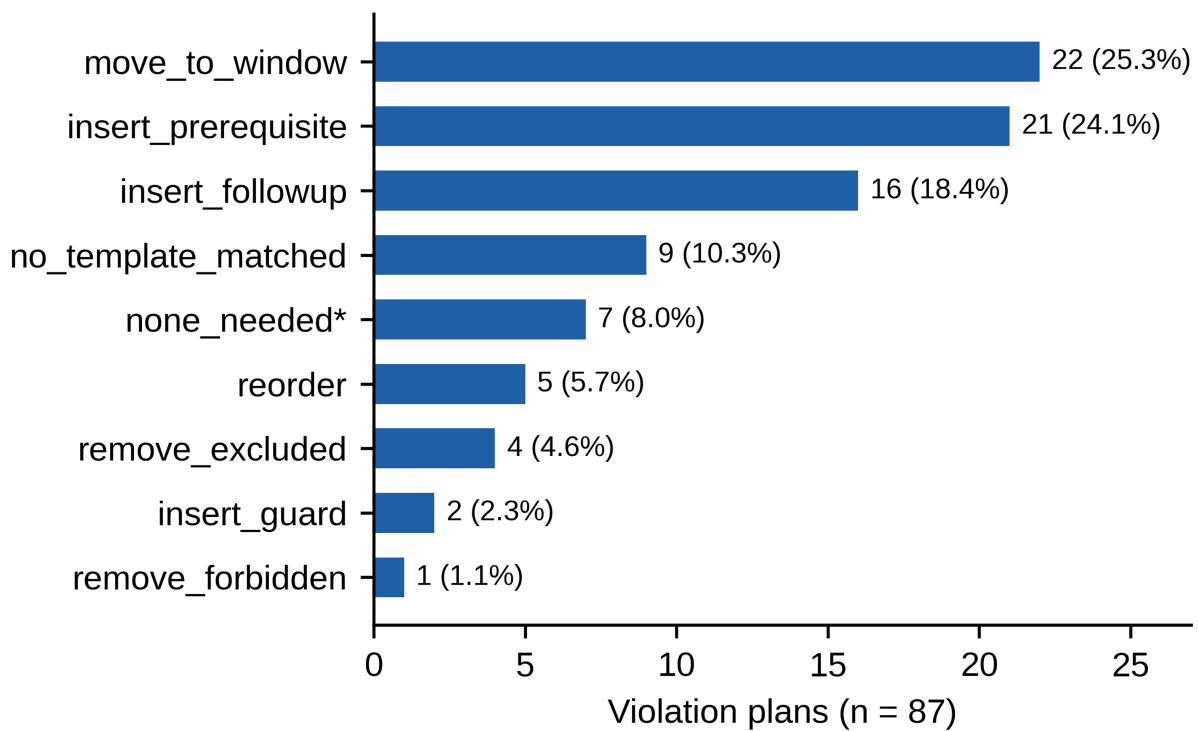


Figure 3. Distribution of repair methods applied across all 87 violation plans (counts and percentages of $n = 87$; one-decimal percentages sum to 99.8% due to rounding). none_needed marks the 7 validator false negatives on which no repair was applied.

Table 5. 2×2 component ablation (F1 scores, aligned set)

	Bag-of-words	Sequential	Δ (seq - BoW)
Handcrafted constraints	0.560	0.652	+0.092
Wiki-IR constraints	0.696	0.920	+0.224
Δ (wiki - craft)	+0.136	+0.268	

Table 6. Template repair by violation type (87 aligned violation plans)

Violation type	Success/Total	Rate	Mean Levenshtein edits (successful)	Failures: cascade/residual
MANDATORY_FOLLOWUP	12/16	75.0%	1.10	2/2
BOUNDED_FOLLOWUP	5/8	62.5%	2.00	2/1
MUTUAL_EXCLUSION	4/8	50.0%	1.00	2/2
FORBIDDEN_SEQUENCE	3/7	42.9%	1.00	4/0
RESPONSE	2/8	25.0%	1.00	0/6
PREREQUISITE	4/20	20.0%	1.00	4/12
ORDERING	4/20	20.0%	2.00	0/16
Overall	34/87	39.1%	1.30 ($n = 27$)	14/39

Edits = mean word-level Levenshtein distance over applied successful repairs (insert/delete = 1, move = 2). Success counts include 7 vacuous cases (validator false negatives; no repair applied); strict success is 27/80 = 33.8%. Repairs target the first detected violation, whose type may differ from the plan's labeled type. Failure causes: cascade = new violations introduced; residual = original violations remain (the residual count includes the 9 ORDERING plans for which no repair template matched and the plan was left unchanged).

pilot scale - 236 plans in total (137 aligned and 99 natural) - and the held-out extraction study, while drawn from a different clinical specialty, remains within healthcare and measures structural rather than semantic transfer; no claim is made beyond this scope, and WikiPlanner should be read as a medical-domain proof of concept within the wider landscape of LLM-based agents^[17] and clinical artificial intelligence (AI)^[18]. Third, natural-set ground truth is generator-assigned and demonstrably noisy (7 of 26 analyzed false negatives trace to label defects); expert re-annotation of the natural set would strengthen future evaluations. Fourth, the extraction audit's inter-annotator agreement is moderate ($\kappa = 0.451$), reflecting a genuinely fuzzy boundary around “implied ordering” text; all disagreements were adjudicated and the calibration process is disclosed, but a larger annotator pool would tighten the estimate. Fifth, extraction recall is not measured: the precision audit and the held-out transfer study both characterize what the extractor produces, not what it misses, and quantifying recall would require complete human gold annotation of a source corpus, which we identify as the highest-value next step. Sixth, the evaluated system grounds actions by exact symbol matching; results for semantically normalized grounding are not claimed and remain future work.

CONCLUSIONS

WikiPlanner demonstrates, on a single medical planning benchmark, that temporal constraints can be automatically extracted from domain wikis and compiled into parallel runtime monitors for validating LLM agent plans. The two-stage TC-IR pipeline - seven labels compiled to six LTLf templates - isolates LLM extraction uncertainty from deterministic compilation and yields transparent, source-localized violation reporting at sub-millisecond latency. We evaluate every stage of that pipeline: extraction precision is 25.9% under dual-annotator audit with adjudication ($\kappa = 0.451$; 79.5% structurally well-formed, so the dominant failure is semantic rather than syntactic), the identical pipeline on a held-out nursing corpus yields 90.1% structurally well-formed records with the same label-scarcity pattern and a 33% silent parse-loss rate from output truncation, aligned-set detection reaches $F1 = 0.920$ with linear latency scaling and a super-additive wiki $\times$ sequential interaction (+0.132), and single-pass repair resolves 39.1% of violation plans with failures dominated by multi-violation residuals and constraint-coupling cascades. The central empirical finding is sharpened rather than softened by the added analyses: the aligned-to-natural gap decomposes into constraint-library coverage and symbol-space disjointness, with the fully grounded handcrafted baseline overtaking the wiki-derived library exactly where grounding is absent. Compiling wiki knowledge into temporal monitors works; grounding the symbols those monitors speak is the problem that decides deployment. Future directions include co-designed action vocabularies, an explicitly evaluated semantic grounding layer, bidirectional temporal windows and exception conditions in TC-IR, expert-annotated natural test sets, iterative multi-pass repair, and - beyond the present scope - replication in non-medical domains.

DECLARATIONS

Acknowledgments

The author thanks the two annotators who performed the TC-IR extraction precision audit reported in this paper, and the adjudicator who resolved the annotation disagreements. Permission to be acknowledged was obtained from all those named above.

Authors' contributions

The author contributed solely to the article.

Availability of data and materials

The evaluated code, the 45-constraint TC-IR library and all intermediate curation stages, the 220 raw extraction records, the 40-symbol action vocabulary, the verbatim extraction prompt and TC-IR schema, both test sets (137 aligned and 99 natural plans) with labels, the case-level repair log and verification workbook, and all analysis and figure sources are openly available in the WikiPlanner repository at <https://github.com/aussie-bzhang/wikiplanner>. Two items are not in the repository: the extraction-audit annotation records and the held-out-corpus extraction outputs summarized in this paper are retained by the author and are available from the corresponding author on reasonable request. No other restrictions apply.

AI and AI-assisted tools statement

During the preparation of this manuscript, the AI tool Anthropic's Claude (Anthropic PBC, model Claude Opus 5) was used to assist with re-analysis of existing experimental logs, drafting and revising manuscript text, generating figures, and preparing the response to reviewers. All reported results derive from the author's own code and data and were verified against the original outputs. All human annotation reported in this paper was performed by human annotators without AI assistance. The tool did not influence the study design, data collection, analysis, interpretation, or the scientific content of the work. The author takes full responsibility for the accuracy, integrity, and final content of the manuscript.

Financial support and sponsorship

None.

Conflicts of interest

The author declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Copyright

© The Author(s) 2026.

Supplementary Materials

[Supplementary Materials](#)

REFERENCES

1. Bai, Y.; Kadavath, S.; Kundu, K.; et al. Constitutional AI: harmlessness from AI feedback. *arXiv* **2022**, arXiv:2212.08073. Available online: <https://doi.org/10.48550/arXiv.2212.08073> (accessed 28 September 2026).

2. Valmeekam, K.; Marquez, M.; Sreedharan, S.; Kambhampati, S. On the planning abilities of large language models - a critical investigation. *arXiv* **2023**, arXiv:2305.15771. Available online: <https://doi.org/10.48550/arXiv.2305.15771> (accessed 28 September 2026).
3. Lewis, P.; Perez, E.; Piktus, A.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv* **2020**, arXiv:2005.11401. Available online: <https://doi.org/10.48550/arXiv.2005.11401> (accessed 28 September 2026).
4. Leucker, M.; Schallhart, C. A brief account of runtime verification. *J. Log. Algebr. Methods Program.* **2009**, *78*, 293-303. DOI
5. Havelund, K.; Roşu, G. Synthesizing monitors for safety properties. In *Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2002*. Katoen, J. P., Stevens, P., Eds.; Springer: Berlin, Germany, 2002; Vol. 2280, pp. 342-56. DOI
6. Kamath A, Zhang S, Xu C, et al. Enforcing temporal constraints for LLM agents. *arXiv* **2025**, arXiv:2512.23738. Available online: <https://doi.org/10.48550/arXiv.2512.23738> (accessed 28 September 2026).
7. Zhang, Y.; Emma, S. Y.; En, A. L. J.; Dong, J. S. RvLLM: LLM runtime verification with domain knowledge. *arXiv* **2025**, arXiv:2505.18585. Available online: <https://doi.org/10.48550/arXiv.2505.18585> (accessed 28 September 2026).
8. Liu, B.; Jiang, Y.; Zhang, X.; et al. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv* **2023**, arXiv:2304.11477. Available online: <https://doi.org/10.48550/arXiv.2304.11477> (accessed 28 September 2026).
9. Kambhampati, S.; Valmeekam, K.; Guan, L.; et al. LLMs can't plan, but can help planning in LLM-modulo frameworks. *arXiv* **2024**, arXiv:2402.01817. Available online: <https://doi.org/10.48550/arXiv.2402.01817> (accessed 28 September 2026).
10. Ji, Z.; Lee, N.; Frieske, R.; et al. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **2023**, *55*, 1-38. DOI
11. Pnueli, A. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, Providence, RI, USA, October 31-November 1 1977; IEEE, 1977; pp. 46-57. DOI
12. De Giacomo, G.; Vardi, M. Y. Linear temporal and dynamic logic on finite traces. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013; pp. 854-60. Available from: <https://www.ijcai.org/Proceedings/13/Papers/132.pdf>. (accessed 2026-09-28).
13. Fuggitti, F. LTLf2DFA [software]. Version 1.0.0.post0. Zenodo; 2019. Available from: <https://zenodo.org/records/3888410>. (Source code: <https://github.com/whitemech/LTLf2DFA>). (accessed 2026-09-28).
14. Rothkopf R, Zeng HT, Santolucito M. Enforcing temporal constraints on generative agent behavior with reactive synthesis. *arXiv* **2024**, arXiv:2402.16905. Available online: <https://doi.org/10.48550/arXiv.2402.16905> (accessed 28 September 2026).
15. Karpathy, A. LLM Wiki. GitHub gist; 6 Apr 2026. Available from: <https://gist.github.com/karpathy/442a6bf555914893e9891c11519de94f>. (accessed 2026-09-28).
16. Team, G. L. M.; Zeng, A.; Xu, B.; et al. ChatGLM: a family of large language models from GLM-130B to GLM-4 all tools. *arXiv* **2024**, arXiv:2406.12793. Available online: <https://doi.org/10.48550/arXiv.2406.12793> (accessed 28 September 2026).
17. Wang, L.; Ma, C.; Feng, X.; et al. A survey on large language model based autonomous agents. *Front. Comput. Sci.* **2024**, *18*, 186345. DOI
18. Rajpurkar, P.; Chen, E.; Banerjee, O.; Topol, E. J. AI in health and medicine. *Nat. Med.* **2022**, *28*, 31-8. DOI PubMed

Disclaimer/Publisher's Note: All statements, opinions, and data contained in this publication are solely those of the individual author(s) and contributor(s) and do not necessarily reflect those of OAE and/or the editor(s). OAE and/or the editor(s) disclaim any responsibility for harm to persons or property resulting from the use of any ideas, methods, instructions, or products mentioned in the content.



© The Author(s) 2026. Open Access This article is licensed under a Creative Commons Attribution 4.0 International License (<https://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, sharing, adaptation, distribution and reproduction in any medium or format, for any purpose, even commercially, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.