

NITI: Neural Plan Concretization for Incremental Execution, Bridging and Trigger Inference from Underspecified Human Policies

Ayan Banerjee

IMPACT Lab, School of Computing
and Augmented Intelligence,
Arizona State University
abanerj3@asu.edu

Shomrik Barua Banerjee

Basis Chandler
Primary North
Chandler, Arizona
banerjeeshormikbarua@gmail.com

Sandeep K.S. Gupta

IMPACT Lab, School of Computing
and Augmented Intelligence
Arizona State University
sandeep.gupta@asu.edu

Abstract

Human decision-making in safety-critical domains is governed by abstract policies that intentionally omit exhaustive preconditions/triggers and contingencies. Executing such underspecified policies reliably in open-world settings remains a fundamental challenge for large language models (LLMs). We introduce NITI, Neural Bridging for Incremental Execution and Trigger Inference from Underspecified Human Policies, a neuro-symbolic framework that treats LLMs not as autonomous planners, but as execution-time concretizers of human intent. NITI incrementally executes abstract policies via verifier-grounded interfaces, infers implicit applicability conditions, repairs execution through neural bridging when assumptions fail, and halts safely under state inconsistency. We evaluate NITI on two structurally distinct embodied domains: a new benchmark of World Cubing Championship 2x2 Rubik’s Cube scrambles (n=50) and a safety-critical automated insulin dosing task. Across multiple frontier LLMs, NITI enables reliable long-horizon execution without task-specific training or search with minimal contextualization inference overhead, substantially outperforming one-shot and chain-of-thought baselines. Our results show that compositional, verifier-grounded execution is essential for safe human-AI collaboration in open-world decision-making. Code and benchmark available here - <https://github.com/ImpactLabASU/ACLNITI>

1 Introduction

Planning and decision-making in embodied AI systems increasingly occur in *open-world, safety-critical environments* such as healthcare, assistive robotics, and human-centered autonomy (Banerjee and Gupta, 2025). In these settings, full state enumeration is infeasible, and agents must operate under partial observability and evolving constraints (Liu et al., 2025). Consequently, **underspecified human policies are the norm rather than the exception** (Dutta et al., 2025). Clinical guidelines and operational procedures encode abstract intent and interfaces but deliberately omit

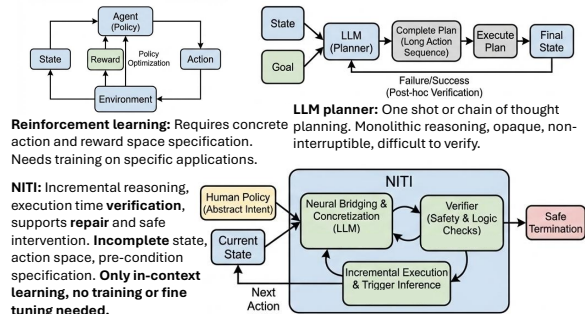


Figure 1: NITI is a framework that can concretize an under-specified human given policy and propose verified safe plans without the need for training or finetuning only using in-context learning.

exhaustive contingencies, relying on the agent to infer context and adapt execution. A natural question is whether these challenges can be addressed through standard Reinforcement Learning (RL) or monolithic Large Language Model (LLM) planning. We argue that both paradigms face structural limitations in this context (Figure 1).

Classical RL provides a general framework for sequential decision making in open-world environments by enabling agents to explore long action sequences and adapt behavior through interaction (Brockman et al., 2016). Exploration-based RL can operate without task-specific rewards by introducing intrinsic objectives such as novelty or prediction error, but still requires a fully specified optimization signal and repeated environment interaction (Pathak et al., 2017). In settings with evolving dynamics or constraints, changes to the underlying Markov decision process necessitate policy relearning or costly re-optimization (Palacios and Geffner, 2016) using time consuming techniques genetic algorithms (Rodriguez-Aguilar et al., 2010), non-linear optimization (Li and Allison, 2017), or constraint satisfaction (D’Ambrosio, 1987) methods. Moreover, RL presupposes a well-defined reward structure and action semantics, making it poorly suited for executing underspecified human policies that encode intent and safety constraints without an explicit objective.

Monolithic LLM planning faces a different but

Table 1: Approaches for planning under underspecified policies. NITI uniquely integrates bridging (B), plan concretization (PC), and trigger inference (TI) through verifier-grounded execution. ✓: support; ×: absence.

Approach	B	PC	TI	Underspecification Resolution	State Transition Type
Interactive Robot Planning (Hori et al., 2024)	✓	×	×	Ask human to clarify ambiguous instructions	Underspec. instruction → Clarified instruction
Introspective Planning (Liang et al., 2024)	✓	×	×	Ask questions when task is ambiguous	Ambiguous task → Disambiguated plan
ExploRLLM (Ma et al., 2024)	✓	×	×	Explore policy to discover learnable regions	Unknown state space → Learnable region
OrionNav (Devarakonda et al., 2024)	✓	×	×	Execute exploration primitives to gather scene info	Incomplete scene → Sufficient scene graph
WorldCloner (Balloch et al., 2023)	✓	×	×	Learn symbolic transitions to adapt after novelty	Unknown dynamics → Learned transitions
Metagent-P (Zhou et al., 2025)	✓	×	×	Replan via symbolic verification/metacognitive reflection	Failed plan → Replanned trajectory
Safe Multi-Robot Planning (Wang et al., 2024)	✓	×	×	Request help when action uncertainty is high	Uncertain action → Assisted action
AIIM (Banerjee et al., 2025)	×	×	✓	Searches for best matched pre-specified plan	Underspecified → Best matching plan
NITI	✓	✓	✓	Generate corrective subplans via verifier feedback	$S_0 \notin \mathcal{D}(\Pi) \rightarrow S' \in \mathcal{D}(\Pi)$

equally severe computational bottleneck. LLMs incur massive training and inference costs precisely because they are *general-purpose* systems. Unlike specialized RL agents, LLMs cannot afford to perform deep, brute-force search over long horizons at inference time. This limitation forces them to sharply limit the depth of action sequences they explore, rendering them ineffective for long-horizon planning. Furthermore, monolithic planning suffers from **non-composability**. When a plan is generated as a single, opaque sequence without well-defined interfaces between sub-tasks, the model capacity required to maintain logical consistency grows exponentially with plan length (Mao et al., 2023; Katz et al., 2025). Monolithic models must simultaneously track long-horizon dependencies, low-level action semantics, and safety constraints within a single context window. As the horizon expands, these models inevitably “forget” constraints or hallucinate transitions, leading to the **concretization gap**: the failure to translate abstract intent into safe, verifiable actions.

Main Contribution: We introduce **NITI** (*Neural Bridging for Incremental Execution and Trigger Inference*), a neuro-symbolic framework that shifts the paradigm from *policy optimization* to **plan concretization** (Figure 1). Unlike traditional planners that require extensive training sets to learn domain dynamics, **NITI requires no parameter updates or finetuning**. It operates entirely through in-context learning, utilizing a **neuro-symbolic tutorial phase** to ground the model in the task’s symbolic structure.

While recent work has explored compositional planning for fully specified domains (Mao et al., 2023), NITI extends this principle to **underspecified domains** where the agent must infer triggers and repair plans on the fly. NITI achieves this through four synergistic mechanisms:

Parameterized Action Schemas: Instead of relying on loose natural language descriptions, NITI utilizes formal action schemas that specify structural transformations. This allows the model to track state changes logically rather than relying solely on semantic probability.

Trigger Inference and Short-Circuiting: In open worlds, the starting state is rarely fixed. NITI dynamically performs *trigger inference* to identify

which policy stage matches the current state. Simultaneously, it employs *structure-aware short-circuiting* to detect when intermediate goals are already satisfied, allowing the agent to skip redundant steps and bypass unnecessary computations.

Verifier-Grounded Operator Repair: When an action fails to produce the expected outcome, NITI does not hallucinate a path forward. Instead, it engages in *verifier-grounded repair*, using feedback to iteratively adjust actions until the target interface is reached or the system halts safely.

Neural Bridging: When the current state falls outside the policy’s applicability domain, NITI generates bridging sequences to transition the system back to a valid state, ensuring robustness against external disturbances.

We evaluate NITI on two distinct domains. First, we introduce a **new benchmark** for symbolic reasoning on the 2×2 Rubik’s Cube, motivated by the limitations of existing datasets such as Sys2Bench (Parashar et al., 2025), which primarily contain short-horizon solutions involving fewer than four moves. Our benchmark consists of 50 official World Cubing Championship (WCA) 2×2 scrambles, each requiring long-horizon reasoning with solution lengths typically exceeding ten moves. Second, we evaluate NITI on a continuous, safety-critical Automated Insulin Delivery (AID) task, comprising 102 open-world planning scenarios derived from a clinical dataset introduced in (Banerjee and Gupta, 2025). Across both domains, results show that by decoupling symbolic structure from neural realization, NITI enables frontier language models to reliably execute underspecified human procedures without task-specific training or finetuning.

2 Related Works

Neuro-symbolic planning has recently gained attention for its ability to support generalization and long-horizon reasoning by decomposing planning into abstract, composable subtasks (Mao et al., 2023; Katz et al., 2025). Symbolic state-action representations enable functional exploration and allow optimal plans to be synthesized at the sub-task level and safely composed for long-horizon execution. Because these abstractions are reusable, they naturally support cross-domain generalization.

While prior neuro-symbolic approaches largely assume fully specified states, actions, and policies, NITI extends this paradigm to open-world settings where states, actions, triggers, and execution conditions are inherently underspecified. Several recent works have focused on real world planning with underspecification with three distinct categories of approaches (Table 1):

Linguistic Clarification: Interactive Robot Action Planning (Hori et al., 2024) and Introspective Planning (Liang et al., 2024) address underspecification by prompting LLMs to reason about uncertainty and request clarification from humans. These approaches assume that underspecification arises from missing linguistic detail and can be resolved by eliciting more complete instructions. In contrast, safety-critical domains often rely on *intentionally abstract* policies—such as clinical guidelines—that omit exhaustive preconditions because full state enumeration is infeasible. NITI departs from clarification-based methods by concretizing abstract human policies through verifier-grounded execution, enabling safe action even when complete specification cannot be provided.

Exploratory Bridging: ExploRLLM (Ma et al., 2024) and OrionNav (Devarakonda et al., 2024) perform bridging via information-gathering actions, using LLM-guided exploration policies or predefined primitives to reduce state uncertainty. Related approaches (Balloch et al., 2023; Zhou et al., 2025) explore unknown states by replanning full trajectories, incurring substantial computational cost. These methods focus on information acquisition rather than *applicability restoration*—returning to states where a policy can be safely executed after assumptions fail. In contrast, NITI generates verifier-grounded corrective subsequences that map a state S_0 not in the policy domain $\mathcal{D}(\Pi)$ to a new state $S' \in \mathcal{D}(\Pi)$, enabling safe continuation of execution rather than mere exploration.

Assistance-Seeking. Safe Task Planning (Wang et al., 2024) uses conformal prediction to detect high action uncertainty and request help from other agents. This approach assumes that external assistance can resolve underspecification, but does not provide mechanisms for autonomous correction or applicability restoration. NITI generates corrective actions autonomously through verifier feedback, enabling execution in safety-critical settings where human oversight may be unavailable.

Partial matching safe plan. AIIM (Banerjee et al., 2025) attempts to match the pre-conditions of an underspecified plan to a fully specified plan already in database. Then it evaluates safety of the queried plan through a simulator, if safe it executes.

NITI’s Contributions: NITI is the first approach to integrate all three capabilities essential for open-

2 × 2 Cube Unrecognized State

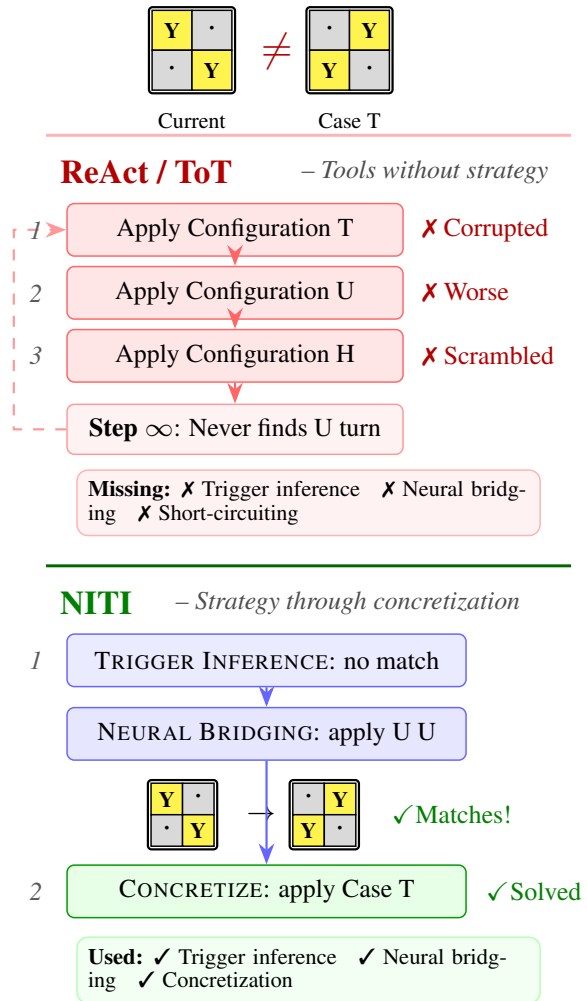


Figure 2: ReAct vs. NITI on an unrecognized cube configuration. ReAct applies algorithms blindly in an infinite loop. NITI performs trigger inference, neural bridging, and plan concretization to solve in two steps.

world execution: **bridging** alone cannot execute abstract policies (which require **concretization**), and neither bridging nor concretization suffice without **trigger inference** to determine when policies apply. By grounding these capabilities in **verifier** feedback rather than human dialogue, or external assistance, NITI enables safe, controllable execution of underspecified human policies in domains where traditional planning assumptions fail.

Distinction from ReAct and Tree of Thoughts: At a surface level, NITI’s incremental execution loop (execute one step → verifier check → correct if error) resembles ReAct (Yao et al., 2023) and Tree-of-Thoughts (Yao et al., 2024). However, this loop is the *scaffolding* of NITI, not its contribution. ReAct and ToT reason about *which* action to take within a fully specified task; NITI reasons about *whether the policy even applies* and what to do when it does not. The mechanisms operating within NITI’s loop, trigger inference, neural

bridging, plan concretization, and structure-aware short-circuiting, have no counterpart in ReAct or ToT. Figure 2 summarizes the structural differences with an example. In Step 6 of the Rubik’s Cube policy (OLL), the agent encounters a yellow diagonal at U[0], U[3] that does not match any taught case. A ReAct agent applies known algorithms (Case T, Case U, Case H) in sequence, each corrupting the state further, because ReAct cannot determine that *no* known trigger matches and cannot transform the state to restore applicability. NITI instead performs **trigger inference** (recognizing no trigger matches), followed by **neural bridging** (applying U rotations to reach a state where Case T applies), then executes the correct algorithm, solving in two steps what ReAct cannot solve at all.

3 Formal exposition of NITI

In this section, we utilize the 2×2 Rubik’s cube (2RC) example to formally represent all components of NITI and codify its operational flow (Figure 3). A 2RC consists of six faces with four colored stickers per face; a scrambled 2RC has mixed colors on each face. The goal is to find a sequence of face turns that returns all stickers on each face to the same color. NITI consists of two phases:

A) TUTORIAL PHASE: This phase utilizes a INTRODUCTION-CHALLENGE-RESPONSE-VERIFICATION cycle to train the LLM agent on the following concepts using in-context learning.

Verifier: V is a domain-specific mechanism that provides legality/ safety checks, and interface evaluations. The verifier may be symbolic or simulation-based. For the 2RC case, we use several verifiers which will be described subsequently.

Symbolic State Space: $S \in \mathcal{S}$ is a structured symbolic state space consisting of typed components with fixed structure and variable bindings. \mathcal{S} does not admit an explicit enumerated transition model. *2RC case:* A cube state is represented as a tuple $U: [U_0, U_1, U_2, U_3]$, $F: [F_0, F_1, F_2, F_3]$, \dots , $D: [D_0, D_1, D_2, D_3]$, where U_i denotes the color of the i^{th} sticker on the upper face, drawn from red, orange, green, blue, white, yellow. Stickers on each face are ordered clockwise starting from the top-right position when viewing that face. The resulting state space and transition set are combinatorially large and infeasible to enumerate within a prompt. Consequently, the LLM is provided with only a partial symbolic description of states and actions, as illustrated in lines 700-721 of the tutorial code (TC) `rubiks_2x2_llama_benchmark.py`. State understanding is evaluated via a corner identification challenge (TC 725–764), with correctness verified by a corner mapping function (TC 235–247).

Parameterized Action Schema: An action schema $a \in \mathcal{A}$ is a symbolic template defining

how components of S are transformed, without enumerating concrete transitions for all $S \in \mathcal{S}$. The effect of an action sequence is inferred through the LLM’s internal logical reasoning rather than explicit transition modeling.

2RC case: The move schema prompt defines actions R, F, L, U, B , and Y , along with their inverses (e.g., R'), by specifying how each move permutes components of the symbolic state. For instance, the R move is described as: On an R move $U[1] \rightarrow B[3]$, $U[2] \rightarrow B[0]$, $F[1] \rightarrow U[1]$, $F[2] \rightarrow U[2]$, $D[1] \rightarrow F[1]$, $D[2] \rightarrow F[2]$, $B[0] \rightarrow D[2]$, $B[3] \rightarrow D[1]$, $R[0] \rightarrow R[1]$, $R[1] \rightarrow R[2]$, $R[2] \rightarrow R[3]$, and $R[3] \rightarrow R[0]$." The inverse is specified succinctly as R' is the inverse of R " (lines 892–904 in the tutorial code). The LLM’s ability to infer action effects is evaluated using a challenge prompt to apply a move sequence to a scramble and requiring the final cube state as output (TC 836–950), with correctness verified via a move-by-move cube mapping function (TC 142–224).

Interface state c_i : It is a state satisfying a given first order logical predicate: p .

Human Policy Structure: A human policy structure Π is a set of abstract under-specified procedural rules specifying allowable actions, sequencing constraints, and interfaces. Π is not required to cover all states in \mathcal{S} and may not always result in a final interface state.

2RC case: The human policy has seven steps each ending in an interface state:

- STEP 1: UNDERSPECIFIED ACTION SEQUENCE - This step has no precondition and applies from any initial state. The LLM is instructed to use an action subsequence $\sigma_1 \in [R', L', RUUR', Y]$ to reach interface c_1 , where $p_1 \Rightarrow D[1] = \text{“white”}$. Because σ_1 may not directly achieve c_1 , the step is under-specified and requires the LLM to explore combinations of these actions (TC 984–998). Step 1 is evaluated on a challenge scramble (TC 1002), with p_1 verified by a state checker (TC 1015). The challenge scramble of the subsequent steps are the parsed cubes from the response of previous step.
- STEP 2: UNDERSPECIFIED SYMBOLIC STATE - This step is conditioned on p_1 and requires selecting one of four action sequences $\sigma_2 = [L'UL, B'U'B, BUB', LUL']$ based on the location of the *white, P, M* piece, where P denotes the sticker facing the observer on the DFR corner and M is unspecified. Successful completion relies on LLM reasoning to reach interface c_2 , satisfying $p_2 \Rightarrow \text{“white, P, M lies in the top layer,“}$ where P and M is left implicit (TC 1033). The step is evaluated via a challenge scramble (TC 1055), with p_2 verified by parsing the cube state from LLM response and applying a state checker (TC 364–381).

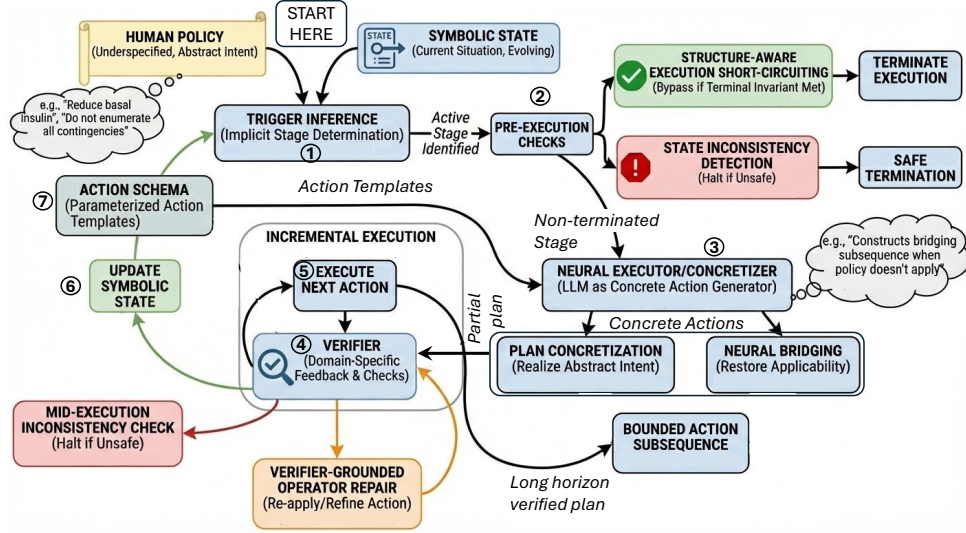


Figure 3: Workflow of NITI through symbolic states, action schema, pre-execution check, neural concretizer, and verified incremental execution.

- STEP 3: UNDERSPECIFIED TRIGGER OR PRE-CONDITIONS - Conditioned on p_2 , this step requires applying a sequence of U, U' turns to reach interface c_3 , satisfying $p_3 \Rightarrow$ “the *white, P, M* piece is at the UFR (U[3], F[1], R[0]) corner.” The appropriate turn sequence depends on the unspecified current location of the piece, requiring the LLM to infer the correct trigger at execution time (TC 1080 challenge TC 1085 verifier TC 384–398).
- STEP 4: LOGICAL EXECUTION OF LONG ACTION SEQUENCE - Conditioned on p_3 , this step requires executing one of four long action sequences selected via natural language preconditions. For example, “if the *white* sticker of the UFR corner is on the top face, apply $UL'U'U'LU'U'L$ ”. Correct execution and state transition must be inferred by the LLM (TC 1113; challenge TC 1118; verifier TC 401-416).
- STEP 5: LOOP UNTIL INTERFACE - The LLM agent is instructed to perform a “Y” turn and then continue from Step 2 through Step 5 until the predicate $p_4 \Rightarrow$ “whole bottom layer solved” is satisfied. What it means for the bottom layer to be solved is implicit (TC 1147, challenge 1151, verifier TC 354-360).
- STEP 6: INCOMPLETE PRE-CONDITION SET - Conditioned on p_4 , this step admits multiple possible triggers, but the prompt specifies long action sequences for only a subset. Implicitly the LLM agent must either identify a matching trigger or apply U, U' turns to reach a state with a known trigger, then execute the corresponding sequence. The checkpoint is $p_5 \Rightarrow$ “the top face solved” (TC 1207; challenge TC 1212; verifier TC 1225).
- STEP 7: INCOMPLETE PRE-CONDITION SET - Conditioned on p_5 , the final step executes a fixed long sequence $RB'RFFR'BRFFRRU$ to solve

Algorithm 1 NITI Execution workflow

Require: Initial state S , human policy steps $\{\Pi_1, \dots, \Pi_K\}$, verifier V
Ensure: Safe execution or termination

- 1: **while** task not finished **do**
- 2: Determine which policy step applies to S {**Trigger inference**}
- 3: **if** S already satisfies a terminal or near-terminal condition or violates safety or consistency rules **then**
- 4: **Stop execution** {**Structure-aware short-circuiting**}
- 5: **end if**
- 6: Generate a short action sequence toward the next interface {**Plan concretization**}
- 7: **for** each action in the sequence **do**
- 8: Execute action with verifier feedback
- 9: **if** action effect is incorrect **then**
- 10: Retry or adjust action until verified
- 11: **end if**
- 12: **If** state becomes unsafe **Stop execution safely**
- 13: **end for**
- 14: **if** policy step is still not applicable **then**
- 15: Generate corrective actions to restore applicability {**Neural bridging**}
- 16: **If** bridging fails **Stop execution safely**
- 17: Execute corrective actions with verifier checks
- 18: **end if**
- 19: **end while**

the cube, though its execution triggers are only partially specified (TC 1243; challenge TC 1249; verifier TC 250–252).

B) EXECUTION PHASE: Successful execution of Step 7 implies that the LLM agent solved its first cube!! This triggers execution phase which uses the following components through Algorithm 1:

Policy Applicability Domain: The applicability domain $\mathcal{D}(\Pi) \subseteq \mathcal{S}$ consists of states for which policy Π prescribes unambiguous actions. In NITI, this domain is not explicitly specified but inferred at execution time from implicit structural and interface conditions.

RC Example: the applicability domain of Step 3 consists of cube states in which the *white, P, M* corner lies in the top layer, a condition never stated explicitly but inferred from execution context.

Incremental Execution: It is an execution regime over $S_t (t \geq 0)$ where, at each step, a finite ac-

tion subsequence $\langle a_t, \dots, a_{t+m} \rangle$ is generated, verified, and applied to move from S_t to an interface state S_{t+m+1} . Subsequence selection is conditioned on the current state, inferred triggers, and verifier feedback, and must satisfy legality and safety at all intermediate states. Execution commits only to bounded subsequences rather than a full trajectory, re-evaluating after each interface is reached.

RC Example: In the tutorial phase, NITI executes each step of the human policy and uses interface verifier to check legality and interface predicate satisfaction after each step.

Plan Concretization: Given an active policy stage Π_k and state S_t , plan concretization constructs at execution time a finite action subsequence $\sigma_t = \langle a_t, \dots, a_{t+m} \rangle$ that respects verifier-enforced legality and safety at all intermediate states and reaches an interface state S_{t+m+1} satisfying the postcondition of Π_k .

RC Example: In Step 4 the abstract instruction “apply the appropriate long sequence based on the orientation of the *white, P, M* corner” is concretized at execution time into the specific action subsequence $\langle U, L', U', U', L, U, L', U', L \rangle$ after inferring the corner and its orientation from the current cube state.

Neural Bridging: Let $\mathcal{D}(\Pi_k) \subseteq \mathcal{S}$ be the applicability domain of policy stage Π_k . Neural bridging constructs at execution time a finite action sequence $\langle a_t, \dots, a_{t+m} \rangle$ that moves the system from $S_t \notin \mathcal{D}(\Pi_k)$ to $S_{t+m+1} \in \mathcal{D}(\Pi_k)$, while satisfying verifier-enforced legality and safety at all intermediate states, without requiring optimality.

RC Example: In Step 6, when none of the explicitly specified triggers matched the current OLL configuration, Claude Opus 4.5 autonomously applied a sequence of U rotations to transform the cube into a state satisfying a known trigger, after which it executed the prescribed long action sequence.

Trigger Inference: Let $\Pi = \{\Pi_1, \dots, \Pi_K\}$ denote ordered policy stages with implicit trigger predicates $\tau_k : \mathcal{S} \rightarrow \{0, 1\}$. Trigger inference computes at execution time $k^*(S_t) = \max\{k \mid \tau_k(S_t) = 1\}$, where the triggers are not explicitly defined but inferred from verifier feedback.

RC Example: when execution begins from a partially solved cube, the LLM infers that the triggers for Steps 2–4 are already satisfied and directly initiates execution from Step 5, despite the policy never stating the conditions under which earlier steps should be skipped.

State Inconsistency Detection: Let \mathcal{I} be domain invariants induced by action schemas \mathcal{A} and policy structure Π , with each $I : \mathcal{S} \rightarrow 0, 1$. State inconsistency detection evaluates whether $\exists I \in \mathcal{I}$ such that $I(S_t) = 0$, indicating a violated invariant, upon

which execution is safely terminated.

RC example: Wrong cube input can be identified by the LLM agent and execution is stopped. Also a correct cube input with one corner is twisted into a rotated configuration rendering the cube unsolvable is also detected early and plan execution is halted.

Verifier-Grounded Operator Repair: Let $a \in \mathcal{A}$ be an action or macro-action with target predicate $p : \mathcal{S} \rightarrow 0, 1$. Verifier-grounded operator repair repeatedly applies a at execution time, producing states $S_{t+i+1} = V.\text{apply}(S_{t+i}, a)$ until $p(S_{t+i}) = 1$ for some $i \geq 1$, or execution is halted due to verifier-detected safety or consistency violations.

RC example: In Step 7, a single execution of the long sequence may not result in a solved cube. The LLM agent infers that the long sequence should be repeated to get the final solution without being prompted to do so.

Structure-Aware Execution Short-Circuiting: Let $\Pi = \Pi_1, \dots, \Pi_K$ be an ordered policy and \mathcal{I}_T a set of terminal or near-terminal invariants $I : \mathcal{S} \rightarrow 0, 1$. Structure-aware execution short-circuiting terminates execution at state S_t when $\exists I \in \mathcal{I}_T$ with $I(S_t) = 1$, as verified, bypassing remaining policy stages.

RC example: Often after performing Step 6 the cube is already in a solved step. The LLM agent should identify this and skip Step 7. This also happens in Step 1, Step 2, and Step 3, where the preconditions for Step 4 or Step 5 are already satisfied. Then the LLM agent should skip Steps and directly execute the most latest step.

4 Applying NITI to AID

Adaptive Insulin Dosing (AID) is a canonical safety-critical open-world execution problem. An AID system continuously delivers basal insulin as automated micro-boluses to regulate blood glucose around a prescribed set point S_p , while enforcing a strict safety constraint against hypoglycemia (blood glucose < 70 mg/dL). In addition to automated control, all FDA-approved AID systems rely on human-initiated insulin actions to handle disturbances such as meals and glucose excursions.

Human intervention primarily occurs through meal boluses u_{meal} , which scale with carbohydrate intake C via a patient-specific carbohydrate–insulin ratio (CIR), while accounting for residual insulin-on-board (IoB). Correction boluses u_{corr} may also be administered when CGM readings exceed S_p , proportional to the deviation using the insulin sensitivity factor (ISF). Accurate IoB estimation is challenging, as it depends on latent insulin pharmacokinetics governed by the Bergman Minimal Model (BMM) (Bergman, 2021) with patient-specific parameters.

Human Policy Representation: In AID, a human-

specified policy defines an abstract sequence of actions—set point adjustments, meal announcements, meal boluses, and correction boluses—intended to satisfy a global safety requirement, typically maintaining glucose above 70 mg/dL. In Signal Temporal Logic (STL), this is expressed as $G \text{ glucose} > 70$. Such policies are considered safe under nominal conditions, for example when evaluated on representative virtual patients from the FDA-approved UVA/Padova Type 1 Diabetes Simulator (Man et al., 2014), and are assessed using aggregate metrics such as time-in-range and time-below-range.

Open-World Under-Specification: These policies are *intentionally underspecified*, omitting complete preconditions, triggers, and corrective actions. Physical exercise, for instance, constitutes an out-of-distribution human-in-the-loop action that alters glucose–insulin dynamics beyond the autonomous capabilities of existing AID systems (MC et al., 2017). Users are instead given high-level guidelines, with execution details left implicit. Longer-term physiological changes, such as pregnancy, further invalidate embedded assumptions and increase manual intervention and user burden (Parent et al., 2023). Even short-term deviations, including skipped or delayed meals, can invalidate planned action sequences.

Symbolic State and Verifier: The execution state is represented symbolically using clinically meaningful variables, including insulin history, glucose trends, meal timing, activity, and safety thresholds, without assuming a fully known or enumerable physiological transition model. BMM dynamics are simulated using the FDA-approved UVA/Padova Type 1 Diabetes Simulator (T1DSim) (Man et al., 2014), which serves as a *verifier* in NITI by evaluating the safety and physiological consistency of proposed interventions.

Incremental Execution and Trigger Inference: Rather than committing to a full dosing plan, NITI executes bounded action subsequences with immediate verifier feedback, enabling adaptation to delayed meals, unexpected exercise, or incomplete information without replanning. Because policy applicability is implicit, the model performs *trigger inference* to determine whether execution should proceed, use a corrective subpolicy, or halt.

Neural Bridging and Execution Repair: If a patient consumes a pre-exercise snack based on CGM readings but does not perform the exercise, the original policy assumption is violated and hypoglycemia risk increases. NITI detects this mismatch and bridges execution by proposing verifier-approved corrective actions, such as temporarily increasing basal insulin, for safe glucose control.

Safe Termination and Deferral: NITI uses T1DSim as a patient-specific verifier, with sim-

Table 2: Stage-wise execution failures across large language models. For each model, all stages to the left of the first failure (F) are successful, and all stages to the right are unsuccessful. Failures correspond to missing NITI execution capabilities. **Legend:** (a) Symbolic state understanding (b) Action Schema Grounding (c) Local Action Sequence Composition (d) Step 1, (e) Step 2 (f) Step 3 (g) Step 4 (h) Step 5 (i) Step 6 (j) Step 7 (k) Verifier-Grounded Plan Concretization (l) Neural Bridging (m) Robust Generalized Execution

Model	Reasoning	a	b	c	d	e	f	g	h	i	j	k	l	m
LLaMA 3.1 / 3.2	No	F	x	x	x	x	x	x	x	x	x	x	x	x
LLaMA 4	No	✓	F	x	x	x	x	x	x	x	x	x	x	x
Phi-4	No	F	x	x	x	x	x	x	x	x	x	x	x	x
Phi4-Reasoning-14B	Yes	✓	✓	F	x	x	x	x	x	x	x	x	x	x
Phi4-Reasoning-Mini	Yes	✓	F	x	x	x	x	x	x	x	x	x	x	x
DeepSeek-R1-8B	Yes	✓	✓	F	x	x	x	x	x	x	x	x	x	x
GPT-o3	Yes	✓	✓	✓	F	x	x	x	x	x	x	x	x	x
GPT-o4-mini	Yes	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GPT-5.2	Yes	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Gemini Pro 3	No	F	x	x	x	x	x	x	x	x	x	x	x	x
Claude Sonnet 4	Yes	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	F	x
Claude Opus 4.5	Yes	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

ulator parameters identified from real patient data using the EMILY technique (Banerjee and Gupta, 2024a,b) following (Banerjee and Gupta, 2025; Banerjee et al., 2025). If long-horizon simulation predicts unsafe outcomes, execution is halted to prevent unsafe continuation.

5 Evaluation

We evaluate several LLM agents on the 2RC case study. LLM agents that pass the 2RC challenge are tested on the AID example.

5.1 Evaluation on the Rubik’s Cube example

Benchmarks: We evaluate performance on two complementary benchmarks.

1. *Sys2Bench Scrambles:* This is the latest large benchmark (Parashar et al., 2025) having 180 test 2RC scrambles. The scrambles are simple and can be solved in less than 5 moves. These demonstrate short term planning objectives which can be efficiently solved by Brute Force or IDA* search.

2. *WCA Scrambles:* Since we want to evaluate NITI on long horizon planning, we introduce a new benchmark consisting of 50 official World Cubing Championship (WCA) 2×2 cube scrambles which have been scrambled with > 10 turns.

Baselines: We compare NITI against 3 baselines:

One-shot prompting: The model is provided with the full cube state and simply asked to solve the cube (*llm_benchmark.py* in supplement).

Chain-of-thought prompting: The model is supplied with the description of Step 1 through 7 of the human policy in one monolithic prompt and explicitly instructed to use the steps (Wei et al., 2022) (*llm_benchmark_cot.py* in supplement).

Program of Thoughts: The model is provided with Step 1 through 7 of the human policy and all verifier code for each step as well as an *apply_move* code (TC 142 - 225) that can

execute the effect of a move and can be recursively applied to get effects of a move sequence. All this is specified in one monolithic prompt (Chen et al., 2023) (`llm_benchmark_pot.py` in supplement). The code for NITI is `rubiks_2x2_llama_benchmark.py`.

Experiments: We perform two experiments: a) Use the tutorial phase of NITI to evaluate which LLM agents could complete the entire tutorial along with corner understanding, action sequence, neural bridging, plan concretization, and at least one full solve (Robust generalized execution, Table 2) challenges discussed in Section 3, and b) of the LLMs which passed experiment (a) we test them on the 50 WCA scramble and Sys2Bench. The evaluation metric is the success rate in solving in each benchmark.

Models evaluated: Table 2 shows all the models that were tested.

Quantitative Results: Experiment (a) failure analysis results are in Table 2. Earlier-generation and smaller models fail at foundational capabilities such as symbolic state parsing or action schema understanding, preventing meaningful evaluation of execution strategies. In contrast, GPT-o4-mini, GPT-5.2, and Claude Opus 4.5 demonstrate competence across all prerequisite reasoning stages.

Across all evaluated models, prompt-only execution regimes fail to generalize to long-horizon cube solving. As summarized in Table 3, one-shot, chain-of-thought (CoT), and program-of-thoughts (PoT) prompting do not solve any WCA scrambles, despite achieving limited success on short-horizon Sys2Bench instances. These results indicate that improved reasoning traces alone are insufficient for sustained execution under underspecified procedures. NITI on the other hand shows significant improvement in solving the 2RC problem with Claude OPUS 4.5 reaching 94% success rate on WCA scrambles. All moves generated by each LLM is provided in attached pdfs.

Failure Mode Analysis. Baseline prompting strategies exhibit systematic structural weaknesses. One-shot prompting often hallucinates invalid action sequences and incorrect intermediate states, while CoT and PoT reduce hallucination but still commit prematurely to complete plans without validation. Consequently, errors propagate unchecked, leading to universal failure on long-horizon tasks.

In contrast, NITI substantially mitigates these failures by deferring commitment to long sequences, validating execution at verifier-grounded interfaces, and halting safely upon inconsistency. Models also infer when early policy stages are already satisfied and correctly transition to later stages unlike prompt-only baselines.

Qualitative Observations. NITI enables execu-

tion behaviors not observed under baseline prompting. Models perform *trigger inference*, identifying when implicit applicability conditions hold despite never being explicitly stated. They also perform *neural bridging*, applying short corrective subsequences (e.g., U turns) to reach states where prescribed operators become valid. Crucially, models detect inconsistent or invalid cube states and terminate safely rather than proceeding with an erroneous plan, an execution-safety property not exhibited by one-shot, CoT, or PoT prompting.

5.2 Evaluation on AID

We evaluate NITI on a safety-critical AID task.

Benchmark. We use the AID benchmark from (Banerjee and Gupta, 2025), consisting of **102 open-world prompts** evenly distributed across three categories: *P1: Novel Actions (NA)*, which introduce out-of-distribution human actions such as unplanned exercise; *P2: Model Adaptation (MA)*, which involve long-term physiological changes (e.g., pregnancy); and *P3: Plan Invalidation (PI)*, which capture short-term deviations such as unexpected food intake. Prompts are expressed in natural language and provide partial contextual information (e.g., CGM values, insulin sensitivity, carbohydrate ratios) while deliberately omitting complete preconditions and corrective instructions. The task is derived from standard Type 1 diabetes insulin management guidelines, which encode abstract safety objectives and timing constraints but intentionally leave many contingencies unspecified. All prompts in supplement.

Baselines. We compare NITI against one-shot prompting, chain-of-thought (CoT), and program-of-thoughts (PoT). One-shot uses the benchmark prompts directly. CoT incorporates clinician exercise-management reasoning from (Riddell et al., 2017). PoT provides a Python implementation of the T1DSim simulator as a reference verifier. We additionally compare against a model predictive control (MPC) baseline similar to Control-IQ, the current state-of-the-art AID system (Resalat et al., 2019).

Metrics. We report: (A) *Safe Execution (%)*, defined as no hypoglycemia (>15 minutes with $CGM < 70, mg/dL$); (B) *Time in Range (TIR, %)*, fraction of CGM samples with $70 < G < 180, mg/dL$; and (C) *Time Below Range (TBR, %)*, fraction of CGM samples $< 70, mg/dL$.

Results. Table 4 summarizes quantitative performance. One-shot prompting frequently produces unsafe recommendations when assumptions are violated. CoT and PoT improve interpretability but do not reliably reduce unsafe behavior due to monolithic reasoning without execution-time verification. In contrast, NITI substantially improves safety and glycemic outcomes across all models and performs

Table 3: Comparison of execution regimes. Under one-shot, chain-of-thought (CoT), and program-of-thoughts (PoT) prompting, no WCA cube scrambles are solved; all successful short solutions occur only in Sys2Bench.

Model	NITI			One-Shot Prompting			Chain-of-Thought			Program-of-Thoughts		
	WCA (%)	Len	Sys2 (%)	WCA (%)	Len	Sys2 (%)	WCA (%)	Len	Sys2 (%)	WCA (%)	Len	Sys2 (%)
GPT-o4-mini	36.0	46 ± 24.1	53.3	0.0	2.7 ± 1	20.0	0.0	2.6 ± 1	33.0	0.0	2.9 ± 1.1	36.0
GPT-5.2	76.0	42 ± 23.7	88.8	0.0	2.4 ± 1	20.0	0.0	2.4 ± 1	16.0	0.0	2.7 ± 1.1	18.0
Claude Opus 4.5	94.0	61.5 ± 16.3	97.8	0.0	2.9 ± 1	33.0	0.0	2.6 ± 1.0	33.0	0.0	3.2 ± 1.1	36.0

Table 4: AID performance under benchmark prompts.

Method	Safe (%)	TIR (%)	TBR (%)
MPC	97	78.2	4.4
One-shot	41.2	61.3	8.7
Chain-of-thought	53.6	69.4	5.2
Program-of-thought	57.2	67.9	4.5
NITI (GPT o4-mini)	86.9	71.9	4.8
NITI (GPT 5.2)	92.4	77.2	3.6
NITI (Claude Opus 4.5)	96.1	81	3.7

Table 5: Failure mode analysis for AID execution. NITI dramatically reduces missed failsafe mode and hallucinated corrective actions.

Method	Hallucinated Fix	Failed to execute fail safe
One-shot prompting	31.6	23.6
CoT prompting	25.7	18.9
PoT prompting	27	11
NITI (GPT o4-mini)	3.9	1.8
NITI (GPT 5.2)	2.1	1.3
NITI (Claude Opus 4.5)	1.1	0.6

on par with the MPC baseline. Claude Opus 4.5 achieves the best overall performance, followed by GPT 5.2 and GPT o4-mini. Notably, all improvements are achieved *without fine-tuning or retraining*, relying solely on verifier-grounded incremental execution and neural bridging.

Failure Mode Analysis: To understand the source of these improvements, Table 5 reports a breakdown of common failure modes. One-shot, CoT and PoT prompting frequently miss fail safe mode triggers, and hallucinate corrective actions. The hallucination examination is performed manually and the protocol is provided in Appendix. NITI sharply reduces all three failure modes by exposing intermediate decision points at which execution can be validated, repaired, or halted.

5.3 Ablation Study

To isolate the contribution of each NITI component, we conduct a systematic ablation on the WCA scramble benchmark, disabling each mechanism individually while keeping all others active. Table 6 reports the results. The component hierarchy is consistent across all three models: **NB** \gg **VR** \gg **TI** \gg **SC**. Removing neural bridging causes the most severe performance collapse (-60 pts for Opus, -54 for GPT 5.2, -24 for o4-mini), confirming that applicability restoration under underspecification is a distinct and critical capability that the remaining components cannot compensate for. Verifier-grounded repair is the second-most critical mechanism (-22 to -40 pts), demonstrating that single-attempt execution without iterative correction fails over long horizons. Trigger inference

Table 6: Ablation study on WCA scrambles (success rate %). Each “-X” column disables that single component. TI = Trigger Inference; NB = Neural Bridging; SC = Short-Circuiting; VR = Verifier-Grounded Repair. “No NITI” reports the best baseline from Table 3.

Model	Full	-TI	-NB	-SC	-VR	No NITI
Claude Opus 4.5	94.0	86.0	34.0	93.0	72.0	0.0
GPT 5.2	76.0	71.0	22.0	64.0	36.0	0.0
GPT o4-mini	36.0	31.0	12.0	32.0	18.0	0.0

has a moderate effect (-5 to -8 pts): when forced to execute sequentially from Step 1 regardless of the current state, models waste moves on already-satisfied stages. Short-circuiting has the smallest impact (-1 to -12 pts), indicating that while useful for efficiency, it is not the primary driver of solve success. Each component addresses a distinct failure mode; no single module subsumes the others, validating the multi-stage design.

Overhead: The tutorial phase requires 35 inference calls, paid once to establish reliable execution primitives. After this one-time setup, each planning task requires only one inference call, as the learned execution scaffolding (interfaces, triggers, verifier interaction) is reused across tasks.

6 Conclusions

We presented **NITI**, a neuro-symbolic framework for executing underspecified human policies in open-world, safety-critical domains. NITI reframes large language models as neural executors that concretize abstract intent via incremental execution, trigger inference, and verifier-grounded repair. Across both symbolic (Rubik’s Cube) and continuous (automated insulin delivery) domains, baseline prompting fails due to premature commitment and lack of execution-time structure. NITI mitigates these failures by introducing composable, verifier-grounded interfaces that enable safe adaptation without search, retraining, or full transition models, and generalize across frontier LLMs using in-context execution alone. These results indicate that failures in symbolic planning stem not from limited model capacity, but from missing execution-time structure. By introducing verifier-grounded interfaces, and neural bridging, NITI transforms LLM planning from brittle monolithic reasoning into composable, interruptible execution of underspecified human policies with limited one time inference overhead in the contextualization step.

Acknowledgements

This work was partially funded by DARPA (AMP, N6600120C4020; FIRE, P000050426), the NSF (FDT-Biotech, 2436801), and the Helmsley Charitable Trust (2- SRA-2017-503-M-B)

References

- Jonathan C. Balloch, Zhiyu Lin, Xiangyu Peng, Mustafa Hussain, Aaron Srinivas, Robert Wright, Julia M. Kim, and Mark O. Riedl. 2023. Neuro-symbolic world models for adapting to open world novelty. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23*, page 2848–2850, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- Ayan Banerjee and Sandeep Gupta. 2024a. [Emily: Extracting sparse model from implicit dynamics](#). In *Proceedings of the 1st ECAI Workshop on "Machine Learning Meets Differential Equations: From Theory to Applications"*, volume 255 of *Proceedings of Machine Learning Research*, pages 1–11. PMLR.
- Ayan Banerjee and Sandeep Gupta. 2025. [Personalized open world plan generation for safety-critical human centered autonomous systems: A case study on artificial pancreas](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 22409–22422, Suzhou, China. Association for Computational Linguistics.
- Ayan Banerjee and Sandeep KS Gupta. 2024b. [Recovering implicit physics model under real-world constraints](#). In *European conference on Artificial Intelligence*, Oral Presentation. arxiv:2412.02215.
- Ayan Banerjee, Aranyak Maity, Imane Lamrani, and Sandeep KS Gupta. 2025. [Towards certified safe personalization in learning-enabled human-in-the-loop human-in-the-plant systems](#). *ACM Journal on Emerging Technologies in Computing Systems*, 22(1):1–27.
- Richard N Bergman. 2021. Origins and history of the minimal model of glucose regulation. *Frontiers in endocrinology*, 11:583016.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI gym. *arXiv preprint arXiv:1606.01540*.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. [Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks](#). *Preprint*, arXiv:2211.12588.
- Bruce D. D'Ambrosio. 1987. Combining symbolic and numeric approaches to uncertainty management. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence*, UAI'87, page 386–393, Arlington, Virginia, USA. AUAI Press.
- Venkata Naren Devarakonda, Raktim Gautam Goswami, Ali Umut Kaypak, Naman Patel, Rooholla Khorrambakht, Prashanth Krishnamurthy, and Farshad Khorrami. 2024. [Orionnav: Online planning for robot autonomy with context-aware llm and open-vocabulary semantic scene graphs](#). *Preprint*, arXiv:2410.06239.
- Subhabrata Dutta, Timo Kaufmann, Goran Glavaš, Ivan Habernal, Kristian Kersting, Frauke Kreuter, Mira Mezini, Iryna Gurevych, Eyke Hüllermeier, and Hinrich Schuetze. 2025. [Problem solving through human-ai preference-based cooperation](#). *Computational Linguistics*, pages 1–36.
- Kazuki Hori, Kanata Suzuki, and Tetsuya Ogata. 2024. [Interactively robot action planning with uncertainty analysis and active questioning by large language model](#). In *2024 IEEE/SICE International Symposium on System Integration (SII)*, pages 85–91.
- Michael Katz, Harsha Kokel, Christian Muise, Shirin Sohrabi, and Sarath Sreedharan. 2025. [Make planning research rigorous again!](#) *Preprint*, arXiv:2505.21674.
- Ruoyu Li and James T. Allison. 2017. [Nonlinear optimization for planning problems with complex constraints and objectives](#). *IEEE Transactions on Automation Science and Engineering*, 14(3):1419–1432.
- Kaiqu Liang, Zixu Zhang, and Jaime Fernández Fisac. 2024. [Introspective planning: Aligning robots' uncertainty with inherent task ambiguity](#). In *The Thirtieth Annual Conference on Neural Information Processing Systems*.
- Xiaotian Liu, Ali Pesaranghader, Hanze Li, Punyaphat Sukcharoenchaikul, Jaehong Kim, Tanmana Sadhu, Hyejeong Jeon, and Scott Sanner. 2025. [Open-world planning via lifted regression with LLM-inferred affordances for embodied agents](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20881–20897, Vienna, Austria. Association for Computational Linguistics.
- Runyu Ma, Jelle Luijkx, Zlatan Ajanović, and Jens Kober. 2024. [ExploRLLM: Guiding exploration in reinforcement learning with large language models](#). In *RSS 2024 Workshop: Data Generation for Robotics*.
- Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli. 2014. The uva/padova type 1 diabetes simulator: new features. *Journal of diabetes science and technology*, 8(1):26–34.
- Jiayuan Mao, Tomás Lozano-Pérez, Joshua B. Tenenbaum, and Leslie Pack Kaelbling. 2023. [What planning problems can a relational neural network solve?](#) In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.

- Riddell MC, Gallen IW, Smart CE, Taplin CE, Adolfsson P, Lumb AN, Kowalski A, Rabasa-Lhoret R, McCrimmon RJ, Hume C, Annan F, Fournier PA, Graham C, Bode B, Galassetti P, Jones TW, Millán IS, Heise T, Peters AL, and 2 others. 2017. [Exercise management in type 1 diabetes: a consensus statement](#). *Lancet Diabetes Endocrinol.*, 5(5):377–390. Epub 2017 Jan 24. Erratum in: *Lancet Diabetes Endocrinol.* 2017 May;5(5):e3. doi: 10.1016/S2213-8587(17)30086-4.
- Héctor Palacios and Hector Geffner. 2016. Learning action models for re-planning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1670–1676.
- Shubham Parashar, Blake Olson, Sambhav Khurana, Eric Li, Hongyi Ling, James Caverlee, and Shuiwang Ji. 2025. [Inference-time computations for llm reasoning and planning: A benchmark and insights](#). Preprint, arXiv:2502.12521.
- Cassandra Parent, Elodie Lespagnol, Serge Berthoin, Sémah Tagougui, Joris Heyman, Chantal Stuckens, Iva Gueorguieva, Costantino Balestra, Cajsa Tonoli, Bérengère Kozon, and 1 others. 2023. Barriers to physical activity in children and adults living with type 1 diabetes: A complex link with real-life glycemic excursions. *Canadian journal of diabetes*, 47(2):124–132.
- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17*, page 2778–2787. JMLR.org.
- Navid Resalat, Wade Hilts, Joseph El Youssef, Nicole Tyler, Jessica R Castle, and Peter G Jacobs. 2019. Adaptive control of an artificial pancreas using model identification, adaptive postprandial insulin delivery, and heart rate and accelerometry as control inputs. *Journal of Diabetes Science and Technology*, 13(6):1044–1053.
- Michael C Riddell, Ian W Gallen, Carmel E Smart, Craig E Taplin, Peter Adolfsson, Alistair N Lumb, Aaron Kowalski, Remi Rabasa-Lhoret, Rory J McCrimmon, Carin Hume, and 1 others. 2017. Exercise management in type 1 diabetes: a consensus statement. *The lancet Diabetes & endocrinology*, 5(5):377–390.
- Juan A. Rodriguez-Aguilar, Ulle Endriss, Sarvapali D. Ramchurn, and Michael Luck. 2010. [Efficient genetic algorithms for multi-agent plan coordination](#). *Journal of Artificial Intelligence Research*, 39:59–101.
- Jun Wang, Guocheng He, and Yiannis Kantaros. 2024. [Safe task planning for language-instructed multi-robot systems using conformal prediction](#). *CoRR*, abs/2402.15368.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS ’22*, Red Hook, NY, USA. Curran Associates Inc.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Yanfang Zhou, Yuntao Liu, Xiaodong Li, Yongqiang Zhao, Xintong Wang, Jinlong Tian, Zhenyu Li, and Xinhai Xu. 2025. [Metagent-P: A neuro-symbolic planning agent with metacognition for open worlds](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22747–22764, Vienna, Austria. Association for Computational Linguistics.

7 Limitations

While NITI demonstrates strong execution fidelity across two challenging domains, several limitations remain.

Dependence on External Verifiers. NITI assumes access to a domain-specific verifier capable of evaluating safety, legality, or interface satisfaction. In Rubik’s Cube, this verifier is exact; in AID, it is a high-fidelity physiological simulator. Constructing such verifiers may be non-trivial in domains where system dynamics are poorly understood or difficult to simulate. Although NITI does not require full transition models, it does rely on partial evaluative feedback, which may limit applicability in settings lacking reliable verification mechanisms.

Scope of Policy Structures. We evaluate NITI on policies that are sequential, staged, and human-authored. While these structures cover many real-world procedural domains (e.g., clinical guidelines, tutorials, operating manuals), NITI is not designed to synthesize policies from scratch or to optimize policies under explicit reward functions. Tasks requiring policy discovery, exploration of novel objectives, or long-term optimization remain outside the current scope.

Computational Cost and Latency. Although NITI avoids search and retraining, incremental execution introduces additional inference calls compared to one-shot prompting. In latency-sensitive applications, especially with large frontier models,

this overhead may be significant. We do not optimize inference efficiency or batching strategies in this work.

Reliance on LLM Reasoning Quality. NITI leverages the reasoning and abstraction capabilities of large language models for trigger inference, plan concretization, and neural bridging. While the framework substantially reduces failure modes, it does not eliminate dependence on model quality. Smaller or weaker models may still struggle with complex symbolic reasoning, particularly in domains with subtle structural constraints.

Human Policy Quality. NITI assumes that the human-specified policy encodes broadly correct intent and safety objectives. If the policy itself is flawed, inconsistent, or unsafe, NITI cannot guarantee correct outcomes and will at best halt execution. Designing tools to detect or repair flawed human policies remains an open problem.

Overall, these limitations highlight that NITI is not a replacement for planning, reinforcement learning, or control, but a complementary execution framework for domains where human intent is known, underspecified, and must be safely realized under uncertainty.

8 Potential Risks

While NITI improves the safe execution of underspecified human policies, several risks remain. First, NITI relies on the quality and fidelity of the verifier: inaccurate or biased simulators may approve unsafe actions or incorrectly halt safe execution. In the AID domain, this risk is partially mitigated by using an FDA-approved simulator, but similar guarantees may not hold in other application areas.

Second, NITI depends on the reasoning capabilities of large language models to infer triggers, construct corrective actions, and interpret symbolic structure. Errors in reasoning—such as misinterpreting state descriptions or action effects—may still occur, particularly under ambiguous or adversarial inputs. Although verifier checks reduce the impact of such errors, they cannot eliminate them entirely.

Third, NITI’s execution-time adaptation may lead to conservative behavior, including frequent halting or deferral, which could reduce system usefulness in time-critical settings. Conversely, overly permissive bridging could mask deeper policy flaws rather than prompting human intervention.

Finally, NITI does not replace domain expertise. Incorrect or poorly designed human-specified policies can still lead to unsafe outcomes, even when executed faithfully. The framework assumes that

high-level policy intent is reasonable and that the verifier encodes appropriate safety constraints.

9 Ethical Considerations

This work addresses safety-critical decision-making domains, particularly healthcare, where ethical considerations are paramount. NITI is explicitly designed as a decision-support and execution framework rather than an autonomous decision-maker. It does not prescribe medical treatment independently and should not be deployed without appropriate clinical oversight.

In the AID setting, all evaluations are conducted in simulation using established, FDA-approved virtual patient models. No real patient data are collected or used, and no clinical decisions are made or recommended for real-world use. The system is not intended to replace clinicians or existing medical devices, but to study execution-time reasoning under underspecified policies.

A key ethical advantage of NITI is its emphasis on safe termination and deferral: when uncertainty or inconsistency is detected, the system halts rather than proceeding with potentially harmful actions. This aligns with the principle of minimizing harm in safety-critical AI systems.

Nevertheless, deploying systems like NITI in real-world settings raises concerns around overreliance, automation bias, and accountability. Clear boundaries must be maintained regarding when human judgment is required, and users must be informed that such systems provide assistance rather than authoritative decisions.

A Appendix

B LLM Prompts

B.1 LLM Prompts for plan generation - GPT o4 mini

[sharp corners, colback=white, colframe=black, boxrule=1pt] **Learn from the following prompts**

Q1: I am eating 30g carbs. Carb ratio is 5. Insulin on board is 3 U. How much bolus should I take?

Answer: You should take 3 U bolus

Q2: I am eating 20g carbs. Carb ratio is 5. Insulin on board is 1 U. How much bolus should I take?

Answer: You should take 3 U bolus

Q3: I am eating 7g carbs to avoid hypoglycemia. Carb ratio is 5. Insulin on board is 2 U. How much bolus should I take?

Answer: You should take 0 U bolus

Q4: I am eating 60g carbs. Carb ratio is 5. Insulin on board is 4 U. How much bolus should I take?

Answer: You should take 8 U bolus

Q5: I am eating 25g carbs. Carb ratio is 5. Insulin on board is 3 U. How much bolus should I take?

Answer: You should take 2 U bolus

Q6: I am eating 7g carbs to avoid hypoglycemia. Carb ratio is 5. Insulin on board is 1 U. How much bolus should I take?

Answer: You should take 0 U bolus
[sharp corners, colback=white, colframe=black, boxrule=1pt] **GPT o4 mini Response**

It seems like you've provided a set of prompts and corresponding answers related to insulin bolus calculations in the context of carbohydrate intake, carb ratio, and insulin on board. Each prompt presents a scenario with specific values, and the corresponding answer provides the recommended bolus insulin amount based on those values.

If you have any specific questions or if there's a particular aspect you'd like me to focus on or elaborate on regarding these prompts, please let me know, and I'll do my best to assist you.

[sharp corners, colback=white, colframe=black, boxrule=1pt] **Prompt** I am eating 45 g carbs. Carb ratio is 5. Insulin on board is 2 U. How much bolus should I take?

[sharp corners, colback=white, colframe=black, boxrule=1pt] **GPT o4 mini Response**

Based on the patterns observed in the provided prompts, the calculation for bolus insulin is determined by the formula: Bolus dose=(Carb intake/Carb ratio)+Insulin on board Substituting in the values: Bolus dose=(45/5)+2=9+2=11 U Therefore, based on the provided information, you should take 11 units of bolus insulin for 45g of carb intake with a carb ratio of 5 and 2 units of insulin on board.

B.2 LLM Prompts for fine-tuning Llama2 Model

[sharp corners, colback=white, colframe=black, boxrule=1pt] **### Instruction:** Find out the diffusion parameter from the Bergman Minimal Model with the following time series. The 40 values corresponding to 400 seconds of IOB values

Input: 1.0 0.99948 0.99747 0.99411 0.98975 0.98473 0.97931 0.97371 0.96808 0.96254 0.95717 0.95205 0.94719 0.94264 0.93839 0.93446 0.93084 0.92752 0.92448 0.92171 0.9192 0.91693 0.91488 0.91303 0.91137 0.90988 0.90855 0.90735 0.90629 0.90534 0.90449 0.90374 0.90307 0.90248 0.90195 0.90148 0.90107 0.90071 0.90038 0.9001

Response: 0.015

B.3 LLM Prompts for final query to fine-tuned Llama2 Model (LLAMA 2)

[sharp corners, colback=white, colframe=black, boxrule=1pt] **Prompt to LLAMA 2**

Below is an instruction that describes the task of finding the Insulin On Board of a type 1 diabetic patient paired with a diffusion parameter of the Bergman Minimal Model for an insulin intake. Write a corresponding output that is the Insulin On Board time series.

Instruction: My diffusion parameter is 0.015081113389040546 and I took an insulin dosage. What is my Insulin On Board percentage time series?

Response:

[sharp corners, colback=white, colframe=black, boxrule=1pt] **Response from LLAMA 2**

Response: 1 0.9992 0.99622 0.9915 0.98568 0.98 0.97398 0.96744 0.9614 0.95591 0.95099 0.94646 0.94225 0.93842 0.93502 0.93198 0.92925 0.92684 0.92469 0.9227 0.92087 0.91922 0.91777 0.91647 0.91526 0.91416 0.91315 0.91222 0.91138 0.9106 0.90987 0.90919 0.90859 0.90803 0.9075 0.90699 0.9065 0.90599 0.90551 0.90505 0.9046 0.90415 0.90372 0.9033 0.90289 0.9025 0.9021 0.90172 0.90137 0.90105 0.90076 0.90049 0.90022 0.900 0.89978 0.89958

C AID verifier

Automated insulin delivery (AID) systems represent a prototypical safety-critical human-in-the-loop, human-in-the-physiology (HIL-HIP) cyber-physical system operating in an open-world setting. These systems continuously administer basal insulin through automated micro-boluses to regulate blood glucose around a prescribed target level S_p , while enforcing a hard safety constraint against hypoglycemia, defined as blood glucose dropping below 70 mg/dL.

In addition to automated control, all FDA-approved AID systems require active human participation to manage exogenous disturbances, most notably food intake. This interaction takes the form of externally delivered insulin doses, referred to as *meal boluses* (u_{meal}), which are computed based on the carbohydrate content C of the meal and a patient-specific carbohydrate-insulin ratio (CIR). Importantly, any residual active insulin remaining in the body—captured by insulin-on-board (IoB)—must be accounted for to avoid overdosing, leading to the formulation in Eq. 1:

$$u_{meal} = C/CIR - iob. \quad (1)$$

Beyond meal-related dosing, patients may administer additional insulin corrections when continuous glucose monitor (CGM) readings exceed

the desired set point. This *correction bolus* (u_{corr}) scales with the deviation between the measured glucose value $G(t)$ and S_p , normalized by the insulin sensitivity factor (ISF), while again compensating for residual insulin activity, as given in Eq. 2:

$$u_{corr} = (G(t) - S_p) / ISF - iob. \quad (2)$$

Accurate estimation of IoB is non-trivial, as it depends on latent insulin pharmacokinetics that are not directly observable by the patient. These dynamics are commonly modeled using the Bergman Minimal Model (BMM) (Bergman, 2021), shown in Eq. 3:

$$\frac{dy}{dt} = z, \quad \frac{dz}{dt} = -2k_1z - k_1^2y + k_1^2u_{ex}, \quad \frac{diob}{dt} = -niob + p_1(y + I_b), \quad (3)$$

where the state vector $X = \{y, z, iob\}$ captures insulin absorption and activity, k_1 governs insulin diffusion, and n and p_1 are patient-specific parameters. The internal variables y and z are latent and cannot be measured directly in practice.

A human-defined safe usage policy for an AID system specifies a sequence of set point adjustments, carbohydrate intake declarations, meal boluses, and correction boluses intended to satisfy a global safety requirement. This requirement is typically expressed as maintaining blood glucose above 70 mg/dL for nearly all time; in Signal Temporal Logic (STL), this constraint can be written as $G \text{ glucose} > 70$, where G denotes the globally-true operator.

For example, a nominal daily policy may prescribe time-specific parameter settings, scheduled meals with corresponding boluses, and conditional correction doses when CGM values exceed predefined thresholds. Such policies are deemed safe under nominal assumptions when evaluated on representative virtual patients from the FDA-approved UVA–Padova Type 1 Diabetes Simulator (Man et al., 2014), which captures population-level physiological variability. Performance is commonly assessed using aggregate clinical metrics, including time-in-range (TIR: $70 \leq \text{CGM} \leq 180$ mg/dL) and time-below-range (TBR: $\text{CGM} < 70$ mg/dL).

C.1 Open-World Failure Modes

P1: Out-of-distribution human actions. Physical exercise introduces acute changes in glucose–insulin dynamics that remain incompletely characterized in clinical models (MC et al., 2017). Consequently, existing AID systems do not autonomously handle exercise events. Instead, users are instructed to follow loosely defined mitigation strategies—such as adjusting set points and consuming a small carbohydrate snack prior to exercise—while post-exercise actions depend on exercise type. These procedures leave timing, applicability, and execution details unspecified.

P2: Long-term physiological drift. Gradual physiological changes, such as those induced by pregnancy, fundamentally alter insulin sensitivity over time and invalidate assumptions embedded in previously safe policies (MC et al., 2017). Maintaining safety under such conditions requires frequent manual adjustments and significantly increases plan management burden (Parent et al., 2023).

P3: Short-term plan violations. Even under otherwise stable conditions, deviations from a planned meal schedule—such as delayed or additional food intake—can invalidate a prescribed action sequence, necessitating immediate corrective reasoning and increasing user workload.

Together, these scenarios highlight that human-authored AID policies are not complete control strategies, but abstract and underspecified plans whose safe execution depends on execution-time interpretation, adaptation, and verification.

D AI Assistant use

Chat GPT 5.2 was used to repair part of broken tutorial code.