# LLM-Augmented Incremental Language Parsing for Robotic Agents

**Mitchell Abrams**[1,*] MITCHELL.ABRAMS@TUFTS.EDU
**Thies Oelerich**[2,*] OELERICH@ACIN.TUWIEN.AC.AT
**Matthias Scheutz**[1] MATTHIAS.SCHEUTZ@TUFTS.EDU
[1] Department of Computer Science, Tufts University, Medford, MA 02155 USA
[2] Automation and Control Institute, TU Wien, Vienna, Austria

## Abstract

Robots that operate with humans must understand language robustly in dynamic, ambiguous, and sometimes noisy environments. Incremental parsing integrated with online motion planning allows robots to adapt their actions in real time as language unfolds, but current systems rely on static grammars and dictionaries, making them brittle when encountering novel or unexpected utterances. We propose a hybrid framework that integrates a large language model (LLM) as a dynamic repair and grammar-adaptation module within a cognitive architecture. When parsing or planning failures arise, the LLM is used to suggest targeted updates to the parser's grammar or lexicon, guided by feedback from downstream components. We implement the incremental parsing and planning modules and describe how the repair system operates using illustrative examples that highlight its potential to expand linguistic competence over time.

## 1. Introduction

As robots become more involved in everyday human environments, they must process spoken language robustly and flexibly, even when instructions are ambiguous, incomplete, or corrected mid-execution. In natural conversations, humans routinely refine or expand their commands incrementally, expecting the robot to adapt its behavior in real time (e.g. "grab the mug... by the handle...and avoid the laptop"). Prior work has combined incremental parsers with online motion planning to achieve this reactive parsing capability (Abrams et al., 2025). However, these systems typically rely on static grammars and dictionaries, making them brittle to novel phrasing, disfluencies, noisy speech, or unexpected corrections. When parsing failures occur, or when ambiguities arise that the grammar cannot resolve, the system may discard valuable partial parses or restart altogether, disrupting the interaction.

Additionally, the language parser is only one component in a cognitive architecture for robot motion control and must be well integrated with the remaining system. Specifically, real-time reactions to natural language corrections require real-time reactions of the task and motion planning system (Abrams et al., 2025). Thus, the design of the language parser must involve interactions with the remaining modules in a cognitive architecture. Furthermore, the language parser needs

0. *These authors contributed equally to this work.

This work is licensed under a Creative Commons Attribution International 4.0 License.

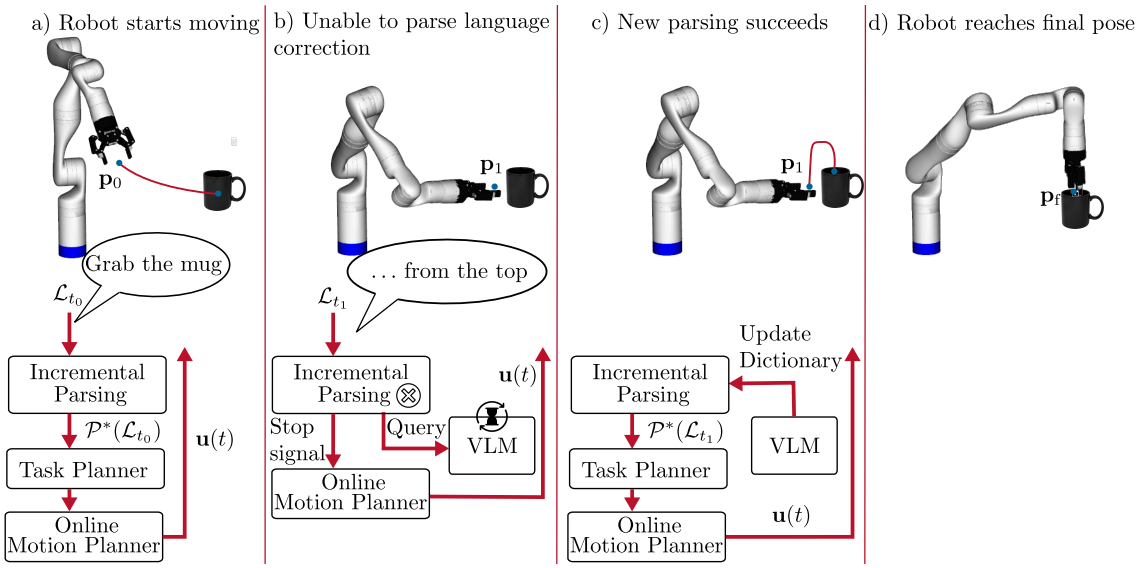


*Figure 1.* Schematic of the LLM-augmented incremental motion planning framework. At time $t_0$ the natural language instruction $\mathcal{L}_{t_0}$ is parsed to $\mathcal{P}^*(\mathcal{L}_{t_0})$. Based on this, the task planner instructs the online motion planner to compute the joint input $\mathbf{u}(t)$. The human instructor utters the language correction $\mathcal{L}_{t_1}$ at time $t_1 > t_0$ to correct the robot to grab the mug from the top instead of the side. However, the parser is unable to parse this correction and sends a stop signal at pose $\mathbf{p}_1$ to the motion planner and queries the VLM for help. The VLM needs time to process the query and then updates the dictionary of the parser, that is then able to create the parse $\mathcal{P}^*(\mathcal{L}_{t_1})$. This parse is handled as before and the robot can start moving again from pose $\mathbf{p}_1$ to the final pose $\mathbf{p}_\mathrm{f}$.

to extract the information relevant to the motion planning (e.g., the planning constraints, from the instruction).

Consider a realistic human-robot interaction scenario, as depicted in Fig. 1. A human says $\mathcal{L}_{t_0}$ = "*Grab the mug,*", which the robot's incremental parser processes successfully and builds the semantic representation $\mathcal{P}^*(\mathcal{L}_{t_0})$, which includes an action (grab) that can be performed on an object (mug) and submits a goal task planner and motion planner to reach for the mug. As the robot begins executing, the user continues: $\mathcal{L}_{t_0}$ = "*... by the bottom.*" At this point, the parser encounters a problem: there is no existing grammar entry for "*by the bottom*" that modifies the grabbing action (e.g., describing the manner of the grabbing strategy), and parsing fails. Rather than discarding the entire parse, the system preserves the valid partial parse for "*Grab the mug*" and triggers a large language model (LLM) or a visual language model (VLM) as a targeted repair module and signals the motion planner to stop. The LLM reasons about the likely meaning of "*by the bottom*," drawing on its prior language knowledge, and suggests dynamically expanding the dictionary with a new rule. This new rule is then parsed to $\mathcal{P}^*(\mathcal{L}_{t_1})$ and send to the motion planner for execution. The parser integrates these updates and resumes parsing from the where it previously failed.

This vignette illustrates how the robot preserves a stable partial parse while locally repairing only the new attachment phrase. The LLM acts as a reasoning bridge between the incremental parser and downstream planning constraints, enabling robust, grounded adaptation to human clarifications

and novel linguistic constructions, all in real time. Such a capability is critical for natural, fluid human-robot collaboration.

In this paper, we present a hybrid architecture, based on the work in (Abrams et al., 2025), that connects an incremental parser, a large language model, and online task and motion planning, through a feedback-driven loop. The LLM is triggered selectively at parse split points, providing targeted dictionary or grammar rule suggestions based on both linguistic plausibility and planning feasibility. In this way, the system incrementally extends its language knowledge, recovers from errors, and resolves ambiguities while preserving stable partial parses and task progress. The LLM is only queried upon parsing failures, at which point the online motion planner is instructed to stop the current motion. This ensures real-time reactivity to natural language instructions and only adds the overhead of an LLM query if parsing fails.

Our main contributions are summarized as follows:
**1)** We propose a novel LLM-based repair mechanism for incremental chart parsing, which leverages split points to provide local, grammar-level corrections. While not yet fully integrated, we illustrate the design and operation of this mechanism with representative examples.
**2)** We implement a system that preserves partial parses by adapting only failing branches, thereby retaining the benefits of real-time incremental interpretation.
**3)** We integrate feedback from online motion planning into the parsing loop, enabling grounded correction suggestions and ensuring physical feasibility and contextual consistency.
**4)** We present a tightly coupled architecture that unifies incremental parsing, planning, and language model-based repair, advancing robust language understanding for grounded HRI.

We implement and demonstrate the incremental parsing and motion planning components of this framework, and illustrate how the proposed LLM-based repair module would operate in realistic robot manipulation scenarios through failure-driven examples and prompt-based outputs.

## 2. Background

### 2.1 Language Understanding

Language understanding for robotic agents poses unique challenges in embodied, real-time interaction (see (Tellex et al., 2020)). A key issue is handling grounded language—utterances referring to real-world objects—and managing natural dialogue with disfluencies, unknown words, and self-corrections. We focus on incremental processing because it flexibly addresses these challenges. Schlangen & Skantze (2011) highlight advantages such as *naturalness* (rapid turn-taking, self-corrections) and *reactivity* (faster, real-time processing). Though discussed in dialogue systems, these benefits are equally critical for embodied agents, where reacting quickly to instructions or mid-utterance revisions (e.g., "grab the box on the right... left") is essential. Incremental processing also aligns with human sentence processing evidence (Trueswell & Tanenhaus, 1995), supporting naturalistic system behavior.

Traditional formalisms like context-free grammar (CFG) (Hopcroft et al., 2001), combinatory categorial grammar (CCG) (Steedman, 2001), and abstract meaning representation (AMR) (Banarescu et al., 2013), as well as statistical parsers, typically produce meaning only after a complete parse, which makes them brittle to online corrections. Abrams et al. (2025) addressed this by de-

veloping an incremental chart parser integrated with a motion planner and reasoning system. The parser incrementally built semantics word-by-word, preserving partial parses while revising others. Yet, its hard-coded grammar rules limited robustness to novel words and constructions.

Statistical methods and LLMs generalize well to novel input (Brown et al., 2020), but sacrifice structured, interpretable representations. Our architecture preserves the incremental chart parser's strengths—maintaining multiple parse hypotheses and enabling pinpoint repairs—while introducing an LLM module for out-of-grammar input. Unlike end-to-end parsing approaches that map raw language directly to actions (Kim et al., 2025; Zhong et al., 2025), we integrate LLMs and VLMs as modular components within a cognitive architecture, retaining symbolic representations while exploiting pretrained models' adaptability.

LLMs and VLMs are increasingly used in robotics, from perception and planning to controls and HRI (Khan & Waheed, 2025). VLMs aid open-world scene description and spatial reasoning, while LLMs generate plans (Ahn et al., 2022) and apply general or social knowledge in unpredictable settings (Mon-Williams et al., 2025). Yet most work bypasses intermediate parsing, instead generating semantic structures or raw text directly. For example, Tuccio et al. (2025) constrain LLM decoding with CFGs, Drozdov et al. (2022) output explicit semantic parses, and Guo et al. (2025) complete constituency trees from partial structures. These operate offline or treat the LLM as a primary decoder, not a repair module for an *incremental* parser in robotics.

Our system differs by invoking the LLM only on failures, proposing local repairs accepted only if validated by a task and motion planner. This feedback loop biases parser choices with feasibility signals, ensuring that language understanding remains grounded in action.

### 2.2 Task and Motion Planning

Task planning creates logical instructions to solve a given task description (Guo et al., 2023). These logical tasks are represented by discrete states to create a sequence of required robot motions (Zhao et al., 2024). The methods range from PDDL (Ghallab et al., 1998) and temporal logic (Takano et al., 2021) to LLM-based reasoning (Oelerich et al., 2024; Chu et al., 2025). A developed task plan is executed by a motion planner. It is advantageous to incorporate feedback from the motion planner to improve task planning, which comprises geometric and dynamic limitations given by the environment (Zhao et al., 2024; Takano et al., 2021).

Motion planning for robotics comprises a multitude of different solutions, including sampling-based approaches (Elbanhawi & Simic, 2014), optimization-based approaches (Oelerich et al., 2025a), and learning-based approaches (Wolf et al., 2025; Dai et al., 2025). In industrial applications, tasks are often repetitive, such that the motion planning information is well known. However, when acting in unknown and dynamic environments, such as households, the motion instructions must be updated during operation, which requires a task planner with a thorough understanding of the environment to update the motion instructions accordingly.

Current approaches to the combined task and motion planning (TAMP) problem include higher-level cognitive systems (Abrams et al., 2025) that interact with the motion planner, or end-to-end planners that reason about the environment and also plan motions (Kim et al., 2025; Zhong et al., 2025). The end-to-end approaches learn mappings between visual and language information to robot actions. This assumes that the trained model understands the language instruction and cor-

rectly correlates it to the visual information to plan suitable motions. Only a single model, without any intermediate representations, is needed for this approach, which makes it computationally efficient to operate once trained. However, training such models is computationally expensive due to the large datasets needed (O'Neill et al., 2024). Furthermore, creating high-quality robotic datasets is challenging and time-consuming. Furthermore, LLMs have been shown to struggle in planning tasks due to hallucinations and long-term planning difficulties (Stechly et al., 2025; Wang et al., 2024). A more modular approach avoids these weaknesses by introducing intermediate representations and limiting the scope of each module. The black-box end-to-end model is replaced by multiple modules that are responsible for solving a certain task, such as object detection (Ravi et al., 2024), task planning (Guo et al., 2023), motion planning (Oelerich et al., 2025a), or language parsing (Abrams et al., 2025). Each of these modules creates intermediate representations that are passed to other modules for further processing until the final module outputs a suitable robot action. This is advantageous to end-to-end approaches in terms of interpretability and safety (Oelerich et al., 2024). Additionally, the modular approaches are less sensitive to small changes in the environment. For example, consider an object grasp operation with a slight change in lighting or object position. This change can lead to considerably different outputs for an end-to-end model, whereas a modular approach, consisting of object detection and point-to-point motion planner, will perform a similar motion as long as the object is detected correctly, decoupling motion planning and detection. However, this comes at the cost of decreased flexibility, as the modules have limited inter-module communication, and a higher computational cost for deploying multiple modules (Oelerich et al., 2024; Kim et al., 2025). As end-to-end and modular approaches have advantages and disadvantages, hybrid approaches are common. In (Oelerich et al., 2024), the language understanding and task planning are performed using black-box models and are combined with visual models and optimization-based motion planning to ensure compliance with constraints. An LLM is used to create PDDL plans in (Chu et al., 2025). The work in (Singh et al., 2023) uses LLMs to create task plans in Python code with underlying motion policies. An LLM is used to create trajectory optimization objectives in Ismail et al. (2024) to instruct a trajectory generator. However, these approaches lack real-time interaction capabilities due to the long inference times of the LLMs.

## 3. Architecture

The architecture of our proposed framework is described in this section. Figure 2 illustrates the relevant components within an existing cognitive architecture, DIARC (Scheutz et al., 2019). DIARC consists of integrate components that contribute to *perceptual information* (speech recognition and vision), *reasoning* (language and planning), and *action* (action execution and speech generation). At a high level this includes a large language model (LLM) component (labeled as LLM/VLM) that works within the system for incremental language understanding and planning.

The system begins with natural language input $\mathcal{L}(t)$ at time $t$, which is processed by an automatic speech recognition (ASR) module and incrementally interpreted using a rule-based grammar, as described in Section 3.1. When parsing failures or ambiguities arise, the LLM component from Section 3.2 is invoked to propose grammar updates or lexical insertions, enabling recovery without discarding prior semantic commitments. Once the parser successfully parses the instruction

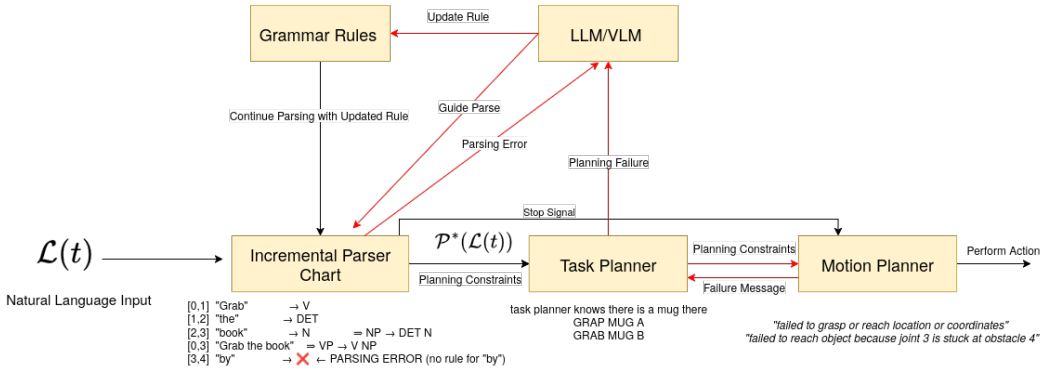


*Figure 2.* Overview of the hybrid architecture combining incremental parsing, LLM-based repair, and task and motion planning.

$\mathcal{L}(t)$, it forwards the planning constraints to the task planner which leverages a motion planner to perform the task, as described in Section 3.3. The task and motion planner report planning failures back to the LLM component, that again proposes a solution. The LLM-enhanced reasoning loop thus enables the system to adapt fluidly to novel or evolving instructions while preserving the transparency and robustness of symbolic interpretation.

### 3.1 Incremental Parser

Our architecture builds upon prior work on incremental language parsing (Abrams et al., 2025), where a robotic agent interprets linguistic input word-by-word and continuously updates its action plans. Figure 2 situates the incremental semantic interpreter within the broader cognitive system. This interpreter maintains a chart-based structure, following the principles of modularity, granularity, and revisability Schlangen & Skantze (2011), allowing it to generate partial interpretations, commit to actions early, and revise prior analyses in light of new information.

The core of the parser is a combinatory categorial grammar (CCG)-style chart parser that stores multiple candidate parses for each span of the utterance. Each word is assigned its lexical category and semantic representation upon arrival. As words accumulate, the parser attempts phrase-level combinations using predefined rules from a grammar file. Once a semantically well-formed constituent is constructed—e.g., "grab the mug"—the system submits this as a preliminary goal $\mathcal{P}^*(\mathcal{L}_t)$ to the motion planner. If the utterance continues (e.g., "...by the top"), the parser extends the chart incrementally, allowing for reinterpretation of the original parse. This incremental approach ensures real-time responsiveness and alignment between language input and robot behavior.

Crucially, our system augments this parser with a large language model (LLM) component. When parsing fails due to missing rules or unknown phrases, the system triggers the LLM to propose targeted dictionary updates. These updates are validated by the downstream motion planner to ensure feasibility, and—if consistent—they are injected into the grammar mid-execution. This allows the robot to recover from unexpected phrasing or out-of-grammar expressions without discarding previous semantic commitments. For instance, a repair from "grab the mug..." to "grab the

mug by the handle" involves extending the grammar to handle the adjunct "by the handle," which is then resolved and integrated seamlessly into the evolving parse.

This architecture enables a tight coupling between language and action: partial parses lead to early execution, while ongoing speech can correct, refine, or constrain the robot's behavior in real time. By dynamically balancing symbolic parsing with LLM-driven repair, the system supports robust and transparent language understanding in open-ended, collaborative HRI scenarios.

### 3.2 LLM-Component

The LLM component acts as a reasoning module that is invoked only on well-defined triggers. It never executes motion directly. Instead, it consumes structured context from other modules and returns proposals that are validated by the symbolic parser and the task and motion planner.

The component is called in two cases: (i) *linguistic parsing events* at split points or failures in the incremental parser (Section 3.1), for example when the grammar lacks an attachment rule; and (ii) *planning failures* reported by the online motion planner (Section 3.3), for example kinematic reach violations or collisions. In both cases the motion planner receives a stop signal, partial parses are preserved, and a compact failure report is created.

The interface is intentionally simple:

```
// Inputs (from parser, planner, belief, VLM)
ParseFailure(span, tokens, local_chart, candidates)
PlannerFailure(reason, object, pose, constraints)
BeliefSnapshot(facts, objects)
SceneGraph(optional, from VLM)

// Outputs (to parser, planner, belief)
GrammarPatch(lexicon_entry | rule)           // to parser
DisambiguationDecision(prefer=VP|NP|referent) // to parser/planner
BeliefUpdate(fact)
```

(1) *Linguistic repair.* On ParseFailure, the component proposes a local *GrammarPatch* (for example a CCG style lexical insertion for a PP modifier) or a *DisambiguationDecision*. The parser injects the patch only if it composes with existing entries, then parsing resumes from the split point. This is the path taken in tables 2 and 3. (2) *Planning informed recovery.* On PlannerFailure, the component reasons over the failure cause and may request a refreshed scene description from the VLM when needed. A recovered parse with re continue the pipeline and submit a new updated goal. The planner validates feasibility before execution. This is illustrated in Listing 2. (3) *Belief maintenance.* When the scene description or language context reveals facts (e.g. objects, properties, or spatial relationships), the component adds or retracts symbolic facts into the knowledge base that downstream modules can use (including the LLM component later on).

The components accesses an LLM (GPT-4) for text-based reasoning and, when necessary, a VLM for scene recognition. VLM queries are slower, therefore they are used sparingly and only after planner feedback indicates that updated perceptual context is needed. By triggering on concrete parser or planner events and by routing proposals back through symbolic validators, the module

remains simple while still providing flexible recovery and grounded reasoning within the overall architecture.

### 3.3 Task and Motion Planning

Robot task and motion planning deals with the problem of planning safe and feasible motions that solve a given task. A language instruction $\mathcal{L}_t$ is parsed at time $t$ and deconstructed by the task planner into sub-tasks, and the motion planner plans a trajectory to solve the sub-tasks. The task planner is not described here further, but many solutions exist, as described in Section 2. In the following, the motion planning problem is described more thoroughly.

#### *3.3.1 Online Motion Planning*

The planning constraints are given by the environment and the robots' physical design, but also by the specified task. Robotic constraints include kinematic and dynamic limits, whereas environmental constraints comprise obstacle avoidance and norm adherence. Of particular importance to this work are the constraints given through the natural language task specifications. These may include many different simple or complex constraints. Examples are the order in which to manipulate objects, certain areas to avoid, certain movements to prefer, or limiting the movement speed. How these constraints are incorporated into the task and motion planning problem depends heavily on the type of constraint. We divide the constraints into types that are only considered in the task planner and constraints that need to be considered in the motion planner. Specifically, kinematic and dynamic constraints need to be considered in the motion planner. Furthermore, constraints that are to be enforced in every time-step, e.g., pose constraints of the end effector, are part of the motion planner. The task planner will deal with the more logical constraints. Most tasks can be split temporally, e.g., when moving a mug, the robot has to grasp it first and then transfer it. This decomposes the task into multiple motion planning problems that are easier to deal with.

The robot is modeled here as a kinematic system with generalized joint states $\mathbf{q}(t)$. The state of the robot is described by the state

$$\mathbf{x}^{\mathrm{T}}(t) = \begin{bmatrix}\mathbf{q}^{\mathrm{T}}(t) & \dot{\mathbf{q}}^{\mathrm{T}}(t) & \ddot{\mathbf{q}}^{\mathrm{T}}(t)\end{bmatrix} \tag{1}$$

and the input to the system is the joint jerk $\mathbf{u}(t) = \dddot{\mathbf{q}}(t)$. The constraints on the systems are handled in the form of constraints on the set of permissible state space $\mathcal{X}(\mathcal{L}_{t_k})$ based on the language instruction $\mathcal{L}_{t_k}$ received at time $t_k$. Based on this definition, the motion planning problem is described as

$$\begin{aligned}
\min_{\mathbf{u}(t)} \quad & c_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t_\mathrm{e}), \mathbf{u}(t_\mathrm{e})) + \int_{t_0}^{t_\mathrm{e}} J_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t), \mathbf{u}(t))\mathrm{d}t \\
\text{s.t.} \quad & \mathbf{x}(t_0) = \mathbf{x}_0 \\
& \mathbf{u}(t_0) = \mathbf{u}_0 \\
& \mathbf{x}(t) \in \mathcal{X}(\mathcal{L}_{t_k}) \quad \forall t : t_0 \leq t \leq t_\mathrm{e} \\
& \mathbf{u}(t) \in \mathcal{U} \quad \forall t : t_0 \leq t \leq t_\mathrm{e}\ ,
\end{aligned} \tag{2}$$

where $\mathbf{x}(t)$ is the robot state, $\mathbf{u}(t)$ is the input and $\mathbf{x}_0$ and $\mathbf{u}_0$ are their initial values at $t_0$. The objective function consists of the terminal cost $c_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t_\mathrm{e}), \mathbf{u}(t_\mathrm{e}))$ and the stage cost $J_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t), \mathbf{u}(t))$,

which are parameterized by $\phi(\mathcal{L}_{t_k})$. The permissible inputs are given by the set $\mathcal{U}$. The formulation (2) works well for single-objective tasks. When the task consists of several sub-tasks, the problem (2) is solved directly for each of the sub-tasks, and a higher-level system sends the appropriate sub-tasks. The above mug example is thus split into two problems of the form (2).

#### *3.3.2 Defining the permissible states*

Defining the state sets (3) may be done in many different ways. We roughly separate the set $\mathcal{X}(\mathcal{L}_{t_k})$ into

$$\mathcal{X}(\mathcal{L}_{t_k}) = \mathcal{X}_{\text{safe}}(\mathcal{L}_{t_k}) \cup \mathcal{X}_{\text{task}}(\mathcal{L}_{t_k}) \cup \mathcal{X}_{\text{robot}} \ , \tag{3}$$

i.e., into the set of safe states $\mathcal{X}_{\text{safe}}(\mathcal{L}_{t_k})$, the set of task-specific states $\mathcal{X}_{\text{task}}(\mathcal{L}_{t_k})$ and the set of valid robot states $\mathcal{X}_{\text{robot}}(\mathcal{L}_{t_k})$

The set $\mathcal{X}_{\text{robot}}(\mathcal{L}_{t_k})$ defines the states that are kinematically feasible for the robot and constitutes, e.g, joint angle, velocity, and acceleration limits. The limits are important to take into account as the robot will otherwise be unable to execute the optimized trajectory (2).

The task set $\mathcal{X}_{\text{task}}(\mathcal{L}_{t_k})$ handles task-specific constraints, e.g., keeping a mug upright or following a path on a surface. These are detrimental for task success. As such constraints are typically defined in the robot's task space, the forward kinematic function is needed to transform the robot state $\mathbf{x}(t)$ into the task-space. This function is generally non-linear, which makes these constraints difficult to handle.

Safety-related constraints are handled by $\mathcal{X}_{\text{safe}}(\mathcal{L}_{t_k})$. These typically include obstacle avoidance for which many different formulations exist (Oelerich et al., 2025b,a). As obstacles are described in the task space these constraints are complicated expressions in the joint space. Furthermore, safety constraints are necessary for human-robot interactions where psychological safety factors (Zacharaki et al., 2020), limit the permissible states to ensure safe interactions.

Additional examples for the mentioned types of constraints are given in Table 1.

| | Safety $\mathcal{X}_{\text{safe}}(\mathcal{L}_{t_k})$ | Task-specific $\mathcal{X}_{\text{task}}(\mathcal{L}_{t_k})$ | Robot $\mathcal{X}_{\text{robot}}(\mathcal{L}_{t_k})$ |
|---|---|---|---|
| Examples | Obstacle avoidance | Upright constraint | Joint angle limits |
| | Motion predictability for humans | Welding path following | Joint velocity limits |
| | Interaction force limits | Timing constraints | Joint acceleration limits |

*Table 1.* Examples of constraints for that limit the set of permissible states $\mathcal{X}(\mathcal{L}_{t_k})$ from (3).

### 3.4 Planning failures

Solving the motion planning problem (2) is complicated due to the nonlinear kinematics of the robot, leading to a non-convex problem. Hence, multiple solutions exist, and the cost function generally has multiple local minima. Moreover, dynamic environments can complicate the problem even further. Due to these difficulties, it can not be expected that the planning problem is solved reliably. Instead, we assume that it is possible that the planning fails and the planner communicates this fail to other modules in the system.

In this work, two different failures are communicated:

**Joint limit:** The robot is stuck in a joint limit, i.e., a joint limit is reached and the joint velocity of all joints is close to zero.
**Constraint limit:** The robot has reached the border of the permissible states $\mathcal{X}(\mathcal{L}_{t_k})$ and the joint velocity of all joints is close to zero.

These failures are not mutually exclusive. Note that a failure is only reported when the robot has reached a limit and does not move anymore, which is detected by checking the constraint values and the joint velocities of the robot. As failure causes are known by the motion planner, these can be efficiently reported to the LLM, which uses the information from the parses and the environment to resolve the issue.

## 4. Interaction Scenarios

We now illustrate how our architecture operates across diverse linguistic challenges in human-robot interaction. Each scenario emphasizes a different interaction failure mode and recovery mechanism. We highlight two examples of constraint feedback from the online motion planner and two examples of the LLM component handling unknown grammar rules and corrective language. These case studies demonstrate the tight coupling between incremental parsing, LLM-based repair, and online motion planning all within real-time language understanding.

### 4.1 Parsing Walkthrough with LLM Repair

| Idx | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | grab | VP → grab | VP → grab the mug | | | **VP → grab the mug by the top** |
| 1 | | the | NP → the mug | | | |
| 2 | | | mug | | | |
| 3 | | | | by | | **PP → by the top** |
| 4 | | | | | the | NP → the top |
| 5 | | | | | | top |

*Table 2.* Parsing chart for "grab the mug by the top".

Table 2 illustrates a parsing walkthrough for the utterance *"grab the mug by the top"*. The indices $[i, j]$ indicate the span of tokens from position $i$ to $j$ in the utterance, and each entry shows the constituent recognized over that span. The system first constructs partial parses incrementally: *"grab"* is inserted at $[0, 1]$ as a verb; *"the mug"* forms a noun phrase at $[1, 3]$; and together, they form a verb phrase *"grab the mug"* at $[0, 3]$. This partial interpretation is immediately submitted to the planner, allowing the robot to begin reaching for the mug.

The next word, *"by"*, triggers a split point. Since the current grammar does not contain an attachment rule for prepositional phrases of this form (e.g., *"by the top"* as a verb modifier), parsing fails at span $[0, 6]$. Rather than discarding the partial parse, the parser retains what it has successfully

built up so far. As the task planner has already received the *"grap the mug"* instruction, it has started moving toward the mug. As there is uncertainty about the correct parse due to the parsing failure, a stop signal is sent to the motion planner. The parsing failure is then passed to the LLM component, which proposes the following candidate rule:

```
C/NN ; #x.graspBy($x, top)
by; PP/PP ; #x.$x
```

This CCG-style rule is validated against planning constraints and, once accepted, integrated into the parser's dictionary in real time (using a dictionary rule injection method). Parsing resumes from the split point, and the phrase *"by the top"* attaches as a verb modifier—contributing the constraint that the robot should grasp the mug at the top. The task planner receives the new instruction and instructs the motion planner with the new goal. This example highlights the preservation of partial parses and their extension through grammar updates.

### 4.2 Attachment ambiguity: "Grab the mug... by the top"

Using the same utterance as input, we assume a different environment– one that leads to a prepositional attachment ambiguity. Table 3 illustrates a parsing walkthrough for the utterance *"grab the mug by the top"*. The final chart cell $(0, 5)$ contains two interpretations: one attaching *"by the top"* to the noun (gray), and one attaching it to the verb (bold). This leads to the interpreations: 1) grab the mug (by means of the top) or 2) grab the mug (located at the top of some location).

The system retains both candidates, signals a stop to the motion planner, and triggers a query to the LLM component to resolve the ambiguity in the environment. The LLM uses information from the belief component (known facts and rules) and the scene description from the VLM together with its internal general reasoning capabilities to resolve the ambiguity. In this case, the LLM component selects the verb-modifying interpretation because there is no "top" location and the planner recognizes that "grasping by the top" is feasible and aligns with the robot's trajectory. The task planner can then instruct the motion planner accordingly and resume movement.

| **Idx** | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| 0 | grab | VP → grab | VP → grab the mug | | | **S → grab (the mug) (by the top)**<br>S → grab (the mug by the top) |
| 1 | | the | NP → the mug | | | |
| 2 | | | mug | | | |
| 3 | | | | by | | **PP → by the top**<br>PP → by the top (modifies NP) |
| 4 | | | | | the | NP → the top |
| 5 | | | | | | top |

*Table 3.* Parsing chart for "grab the mug by the top" showing an attachment ambiguity.

### 4.3 Reference resolution with planner-informed LLM recovery: "Hand me the screwdriver"

A human instructs an agent: *"hand me the screwdriver."*. In this case, there is no attachment ambiguity, but a referential ambiguity. There are two screwdrivers in the scene, but only one is practically reachable. The system begins with the most salient candidate, then reacts to a planner failure by handing control to the LLM component for a quick scene re-description and goal resubmission.

*Listing 1.* Parsing chart (simplified) for "hand me the screwdriver":

```
Utterance: "hand me the screwdriver"

[0,1] hand            -> V
[1,2] me              -> NP (pron)
[2,4] the screwdriver -> NP
[0,2] VP              -> hand me
[0,4] VP              -> hand me the screwdriver
```

*Listing 2.* World state and constraints for screwdriver handover

```
// Initial belief (before LLM refresh)
screwdriver_1 : type = screwdriver, pose = shelf_left,  reachable = false
// screwdriver_2 not yet selected as a candidate

// Goal (from VP semantics)
goal: bring_to(human, screwdriver_k)

// Planner check
precondition: reachable(screwdriver_k)

// Step A: try screwdriver_1
reachable(screwdriver_1) = false
-> planning failure: reason = kinematic_reach_violation

// Failure handoff to LLM component
LLM input:
  - failure(reason = kinematic_reach_violation, object = screwdriver_1)
  - current belief facts and camera view
LLM action:
  - request fresh scene semantics (updated scene graph)
  - identify alternative screwdriver with reachable-looking pose

// Belief update from LLM scene description
screwdriver_2 : type = screwdriver, pose = bench_right, reachable = true

// Step B: resend same goal with updated candidate set
reachable(screwdriver_2) = true
collision_free_path(screwdriver_2) = true
-> plan found, execute handover
```

The planning chart is given in Listing 1 and the evolving actions are described in Listing 2. After the VP at $[0, 4)$ is built, the referent resolver proposes *screwdriver_1* first. The online motion planner

rejects it with a kinematic reach violation and returns a structured failure. This failure is forwarded to the LLM component, which requests an updated scene description, identifies *screwdriver_2* in a reachable-looking pose near *bench_right*, and updates the belief state. The system re-submits the same goal, the planner validates reachability and a collision-free path, and the motion planner executes the handover with *screwdriver_2*. The LLM is able to resolve the failure and guide the robot to a successful task execution while providing interpretable outputs to the other modules.

*Obstacle-induced failure variant*

*Listing 3.* World state and constraints for screwdriver handover with obstacles

```
// Objects
screwdriver_1 : type = screwdriver, location = shelf_left,
                reachable = true,  blocked_by = glass_cup
screwdriver_2 : type = screwdriver, location = bench_right,
                reachable = true,  blocked_by = none

// Goal (from VP semantics)
goal: bring_to(human, screwdriver_k)

// Planner checks (online)
precondition:
  reachable(screwdriver_k)            // kinematic reach
  and collision_free_path(screwdriver_k) // obstacle-free motion

// Evaluation
reachable(screwdriver_1) = true
collision_free_path(screwdriver_1) = false  -> candidate rejected
reachable(screwdriver_2) = true
collision_free_path(screwdriver_2) = true   -> candidate accepted

chosen referent: screwdriver_2
```

We consider the same instruction as above, parsed in Listing 1, but the *screwdriver_1* is unreachable due to an obstructing obstacle. The motion planner returns a failure due to the obstruction but is not kinematically infeasible. This is described in detail in Listing 3. As the parser completes the VP at $[0, 4]$ (*"hand me the screwdriver"*), the referent resolver proposes candidate screwdrivers. The online planner enforces both kinematic reachability and a collision-free path. Although *screwdriver_1* is reachable, the path is obstructed by a *glass_cup*, so the motion plan fails the collision check. *screwdriver_2* satisfies both checks, so the system commits to *screwdriver_2* and proceeds with the handover.[1]

1. If the task policy permitted clearing obstacles, the planner could propose a subgoal (e.g., *move(glass_cup)*), but in this example we assume direct handover without environment rearrangement.

## 5. Discussion & Conclusion

Our proposed framework demonstrates that combining incremental parsing with LLM-augmented repair inside a cognitive architecture can improve robustness in human-robot interaction (HRI). By leveraging partial parses and only invoking the LLM when failures occur, the system balances symbolic transparency with neural adaptability. This approach preserves interpretability and enables recovery from unexpected linguistic phenomena such as novel phrasing, corrections, or referential ambiguity. Importantly, these are core challenges that will be encountered in human-robot dialogue.

Since the LLM is allowed to extend grammar rules without strict logical checks, it may propose rules that are nonsensical or ill-formed. While such rules rarely compromise safety, because downstream motion and task planning enforce feasibility through kinematic and collision constraints, they can reduce linguistic coherence or lead to failures in the task planner. However, if a proposed grammar rule fails, it will also trigger the LLM component again for another repair. Another limitation concerns ambiguity resolution: in some cases, only perceptual grounding can disambiguate instructions (e.g., selecting between visually identical objects). We addressed this by delegating this task to a vision–language model (VLM), but future extensions could incorporate more structured perceptual pipelines, such as object detection and scene graphs, to provide richer grounding.

Beyond these considerations, the broader contribution lies in how the cognitive architecture enables the use of a traditional incremental chart parser in modern HRI. On its own, a static grammar-based parser is too brittle for deployment in open, noisy environments. At the same time, purely statistical or LLM-based approaches sacrifice interpretability and reliability. Our architecture bridges this gap: the parser provides structure and transparency, the LLM supplies targeted repairs and generalization, and the task/motion planner enforces feasibility and safety. In effect, the architecture makes chart-based incremental parsing a viable and scalable strategy for robotics, even in the era of large language models.

In conclusion, this work highlights the value of embedding LLMs not as replacements for symbolic parsing, but as repair and reasoning modules within a cognitive system. Future work will explore extending this approach to richer perceptual grounding and handling multiple failures (e.g., multiple motion planning failure types at once). Taken together, these directions outline a path toward robots that can robustly understand and act on human instructions while retaining transparency, safety, and adaptability.

## References

Abrams, M., Oelerich, T., Hartl-Nesic, C., Kugi, A., & Scheutz, M. (2025). Incremental Language Understanding for Online Motion Planning of Robot Manipulators. *Preprint arXiv:2508.06095*.

Ahn, M., et al. (2022). Do as i can, not as i say: Grounding language in robotic affordances. *Preprint arXiv:2204.01691*.

Banarescu, L., et al. (2013). Abstract meaning representation for sembanking. *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse* (pp. 178–186).

Brown, T., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Chu, K., Zhao, X., Weber, C., & Wermter, S. (2025). LLM+MAP: Bimanual Robot Task Planning using Large Language Models and Planning Domain Definition Language.

Dai, X., Yang, Z., Yu, D., Zhang, S., Sadeghian, H., Haddadin, S., & Hirche, S. (2025). Safe Flow Matching: Robot Motion Planning with Control Barrier Functions. *Preprint arXiv:2504.08661*.

Drozdov, A., Schärli, N., Akyürek, E., Scales, N., Song, X., Chen, X., Bousquet, O., & Zhou, D. (2022). Compositional semantic parsing with large language models. *Preprint arXiv:2209.15003*.

Elbanhawi, M., & Simic, M. (2014). Sampling-Based Robot Motion Planning: A Review. *IEEE Access*, *2*, 56–77.

Ghallab, M., Howe, A., McDermott, D., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). *PDDL-The Planning Domain Definition Language*. Technical Report CVC TR-98-003/DCSTR-1165, Yale Center for Computational Vision and Control.

Guo, H., Wu, F., Qin, Y., Li, R., Li, K., & Li, K. (2023). Recent Trends in Task and Motion Planning for Robotics: A Survey. *ACM Comput. Surv.*, *55*, 289:1–289:36.

Guo, P., Zhang, M., Li, J., Zhang, M., & Zhang, Y. (2025). Contrastive learning on llm back generation treebank for cross-domain constituency parsing. *Preprint arXiv:2505.20976*.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, *32*, 60–65.

Ismail, S., Arbues, A., Cotterell, R., Zurbrügg, R., & Alonso, C. A. (2024). NARRATE: Versatile Language Architecture for Optimal Control in Robotics. *Preprint arXiv:2403.10762*.

Khan, M. T., & Waheed, A. (2025). Foundation model driven robotics: A comprehensive review. *Preprint arXiv:2507.10087*.

Kim, M. J., et al. (2025). OpenVLA: An Open-Source Vision-Language-Action Model. *Proceedings of the Conference on Robot Learning* (pp. 2679–2713).

Mon-Williams, R., Li, G., Long, R., Du, W., & Lucas, C. G. (2025). Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, (pp. 1–10).

Oelerich, T., Beck, F., Hartl-Nesic, C., & Kugi, A. (2025a). BoundMPC: Cartesian path following with error bounds based on model predictive control in the joint space. *The International Journal of Robotics Research*, *44*, 1287–1316.

Oelerich, T., Hartl-Nesic, C., Beck, F., & Kugi, A. (2025b). BoundPlanner: A convex-set-based approach to bounded manipulator trajectory planning. *Preprint arXiv:2502.13286*.

Oelerich, T., Hartl-Nesic, C., & Kugi, A. (2024). Language-guided Manipulator Motion Planning with Bounded Task Space. *Proceedings of the Conference on Robot Learning*.

O'Neill, A., et al. (2024). Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration0. *Proceedings of the International Conference on Robotics and Automation* (pp. 6892–6903).

Ravi, N., et al. (2024). SAM 2: Segment Anything in Images and Videos. *Preprint: arXiv:2408.00714*.

Scheutz, M., Williams, T., Krause, E., Oosterveld, B., Sarathy, V., & Frasca, T. (2019). An overview of the distributed integrated cognition affect and reflection diarc architecture. *Cognitive architectures*, (pp. 165–193).

Schlangen, D., & Skantze, G. (2011). A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*, *2*, 83–111.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., & Garg, A. (2023). ProgPrompt: Program generation for situated robot task planning using large language models. *Autonomous Robots*, *47*, 999–1012.

Stechly, K., Valmeekam, K., & Kambhampati, S. (2025). On the self-verification limitations of large language models on reasoning and planning tasks. *Proceedings of the International Conference on Representation Learning* (pp. 98190–98243).

Steedman, M. (2001). *The syntactic process*. MIT press.

Takano, R., Oyama, H., & Yamakita, M. (2021). Continuous Optimization-Based Task and Motion Planning with Signal Temporal Logic Specifications for Sequential Manipulation. *Proceedings of the International Conference on Robotics and Automation* (pp. 8409–8415).

Tellex, S., Gopalan, N., Kress-Gazit, H., & Matuszek, C. (2020). Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, *3*, 25–55.

Trueswell, J. C., & Tanenhaus, M. K. (1995). Sentence comprehension. *Handbook in perception and cognition*, *11*, 217–262.

Tuccio, G., Bulla, L., Madonia, M., Gangemi, A., et al. (2025). Grammar-llm: Grammar-constrained natural language generation. *Findings of the Association for Computational Linguistics: ACL 2025* (pp. 3412–3422).

Wang, L., et al. (2024). A Survey on Large Language Model based Autonomous Agents. *Frontiers of Computer Science*, *18*, 186345.

Wolf, R., Shi, Y., Liu, S., & Rayyes, R. (2025). Diffusion Models for Robotic Manipulation: A Survey.

Zacharaki, A., Kostavelis, I., Gasteratos, A., & Dokas, I. (2020). Safety bounds in human robot interaction: A survey. *Safety Science*, *127*, 104667.

Zhao, Z., Cheng, S., Ding, Y., Zhou, Z., Zhang, S., Xu, D., & Zhao, Y. (2024). A Survey of Optimization-Based Task and Motion Planning: From Classical to Learning Approaches. *IEEE/ASME Transactions on Mechatronics*, (pp. 1–27).

Zhong, Y., et al. (2025). DexGraspVLA: A Vision-Language-Action Framework Towards General Dexterous Grasping. *Preprint: arXiv:2502.20900*.