Autonomous Multi-Robot Action Planning Through Controlled Robot Language

Dang Tran[1], Minhazul Arefin[1], Zhengchen Zhang[2], and Hongsheng He[1](✉)

[1] The University of Alabama, Tuscaloosa, AL 35487, USA
hongsheng.he@ua.edu

[2] Infocomm Technology Cluster, Singapore Institute of Technology, Singapore, Singapore

Abstract. Interacting with multiple robots presents significant challenges for non-experts, especially in systems requiring concurrent behaviors. This paper introduces a plan generation method for multi-robot systems using Controlled Robot Language. The framework generates planning scripts with temporal constraints from large complex instructions, incorporating contextual awareness and resolving conflicts through mutual exclusivity expressions. Subjects, objects, and action parameters are extracted from the instructions and are grounded into robotic actions. Through linguistic evaluations and real-time experiments, the framework demonstrates its robustness in language comprehension and the ability to capture and provide temporal solutions for multi-robot planning. This work contributes to the accessibility and usability of multi-robot communication for non-experts using natural language.

Keywords: Planning · Natural Language Processing · Human Robot Interaction · Multiple Robot System

1 Introduction

Multiple robot planning, inspired by biological systems, addresses complex tasks beyond individual robot capabilities. However, interacting with these systems often requires expertise, limiting accessibility for non-experts. In this paper, we propose a communication framework using natural language commands to enable effective interaction with a multi-robot team. This approach aims to enhance accessibility for non-experts in multi-robot controls, bridging the gap between robotic teams and human operators.

Communicating with multi-robot teams using natural language presents challenges beyond typical linguistic platforms. Besides addressing language ambiguities, expressivity, and domain generality, multi-robot planning interfaces must

This work was supported by NSF 2327313 and Alabama EPSCoR Graduate Research Scholars Program Fellowship.

accommodate concurrency and mutual awareness. Concurrency allows simultaneous actions for complex operations, which enables multiple robots to collaborate. Mutual awareness, on the other hand, allows robots to understand the environment of itself and other robots, to avoid conflicts during planning and execution. These requirements significantly increase the complexity of multi-robot planning compared to single-robot domains.

While linguistic communication for individual robots has been explored [1,4], multi-robot communication remains a challenge. As the number of agents increases, the existing methods struggle to interpret instructions. This highlights the need for new approaches for multi-robot communication. Multi-robot communication was first addressed using the Generalized Grounding Graph approach, which can directly convert linguistic symbols into robotic actions [7]. However, the method primarily recognizes symbolic tokens, and overlooks syntactic and semantic structures, limiting its use in complex planning. To address this issue, a temporal logic framework for complex planning problems was proposed [6]. However, the method requires formal syntax inputs and a deep understanding of temporal logic. Additionally, temporal constraints are described in these methods using programming syntax, which is not intuitive for non-expert operators.

In this paper, we proposed a linguistic-based plan generation framework for multi-robot teams, using Controlled Robot Language (CRL). Extending from our prior work for individual robot platforms [8,9], this approach generates deterministic planning scripts from complex large-scale commands. The framework incorporates sophisticated capabilities to describe temporal constraints and mutual exclusivity (mutex) expressions. The generated plan is complete and comprehensible by temporal solvers, which eliminates the need for extensive postprocessing or human intervention. The contributions of this paper are:

1. We developed a CRL-based plan generation framework for multi-robot teams, which can comprehend complex, large-scale linguistic instructions. The proposed framework addresses critical requirements of concurrency and contextual awareness in multi-agent planning.
2. We demonstrated the robustness and applicability of the framework on real-time multi-robot simulation, emphasizing the need of context-aware communication methods for effective multi-robot navigation.

2 Plan-Generation for Multi-Robot Team

The general workflow of CRL-based framework for plan generation is visualized in Fig. 1. Given large contextual instructions in natural language, the framework returns deterministic semantics in the form of discourse representation structure (DRS). Information blocks are extracted from semantics, including temporal constraints and mutex expressions. The extract semantics are used to construct a complete Planning Domain Definition Language (PDDL) planning script, which is directly interpretable by the temporal solvers, and triggers the corresponding controls on each robot.

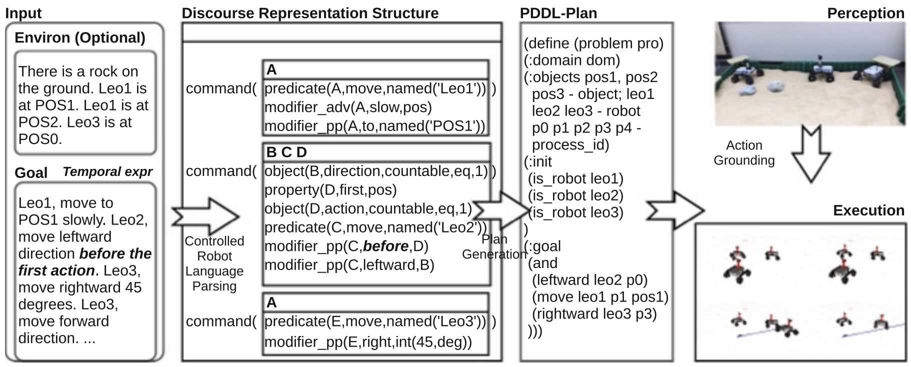

Input
Environ (Optional)
There is a rock on the ground. Leo1 is at POS1. Leo1 is at POS2. Leo3 is at POS0.
Goal
Temporal expr
Leo1, move to POS1 slowly. Leo2, move leftward direction before the first action. Leo3, move rightward 45 degrees. Leo3, move forward direction. ...
Controlled Robot Language Parsing
Discourse Representation Structure
command(predicate(A,move,named('Leo1')) modifier_adv(A,slow,pos) modifier_pp(A,to,named('POS1')))
command(object(B,direction,countable,eq,1) property(D,first,pos) object(D,action,countable,eq,1) predicate(C,move,named('Leo2')) modifier_pp(C,before,D) modifier_pp(C,leftward,B))
command(predicate(E,move,named('Leo3')) modifier_pp(E,right,int(45,deg)))
Plan Generation
PDDL-Plan
(define (problem pro)
(:domain dom)
(:objects pos1, pos2 pos3 - object; leo1 leo2 leo3 - robot p0 p1 p2 p3 p4 - process_id)
(:init (is_robot leo1) (is_robot leo2) (is_robot leo3))
(:goal (and (leftward leo2 p0) (move leo1 p1 pos1) (rightward leo3 p3))))
Perception
Action Grounding
Execution

Fig. 1. CRL-based plan generation framework for multi-robot control.

2.1 Controlled Robot Language Parsing

A multi-layer CRL pipeline [8] is applied to construct semantics from the large-scale linguistic inputs. CRL can provide both syntactic and semantic analysis of the natural language. Developed using over 300 Context Free Grammar rules, CRL syntax covers lots of natural language expressions used in human-robot interaction. The CRL contains multiple preprocessing layers to analyze syntax, detect typographical errors, and automatically translate original instructions into a more suitable format that CRL semantic parser can understand. Given valid CRL-syntax instructions, CRL semantic parser generates a deterministic formal logic in linear DRS. Figure 2 illustrates the details of CRL, which contains seven layers: a tokenizer, lemmatizer, POS tagger, phrase chunker, syntactical parser, translator, and semantic parser.

Given the semantics in linear DRS, we extract core information using the Bottom-Up traversal algorithm. The linear DRS (Fig. 3a) is first translated into tree-like semantic structure (Fig. 3b) using a DRS parser. The Bottom-Up algorithm initiates at the root nonterminal and proceeds in a depth-first manner. During the iteration, symbolic tokens at specific nodes are captured locally and

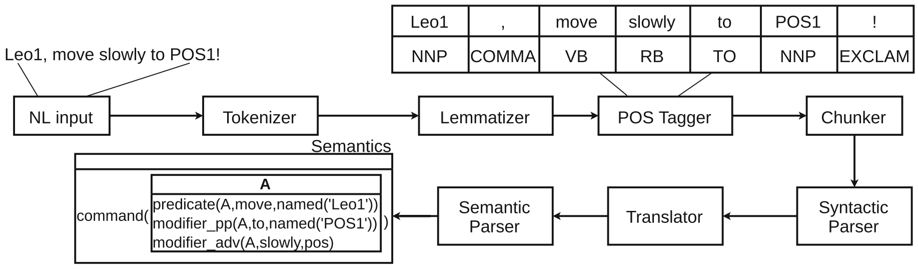

Leo1, move slowly to POS1!
NL input
Tokenizer
Lemmatizer
POS Tagger
Chunker
Leo1 | , | move | slowly | to | POS1 | !
NNP | COMMA | VB | RB | TO | NNP | EXCLAM
Syntactic Parser
Translator
Semantic Parser
Semantics
command(predicate(A,move,named('Leo1')) modifier_pp(A,to,named('POS1')) modifier_adv(A,slowly,pos))

Fig. 2. Multi-layer CRL transformation: From natural language to formal semantic representation.

propagated back to their parents. The captured tokens are encapsulated with variable IDs and organized into information plates, representing partial data. Each information plate captures a specific property of an event. To capture the full context semantics, these plates are merged using ID cross-referencing.

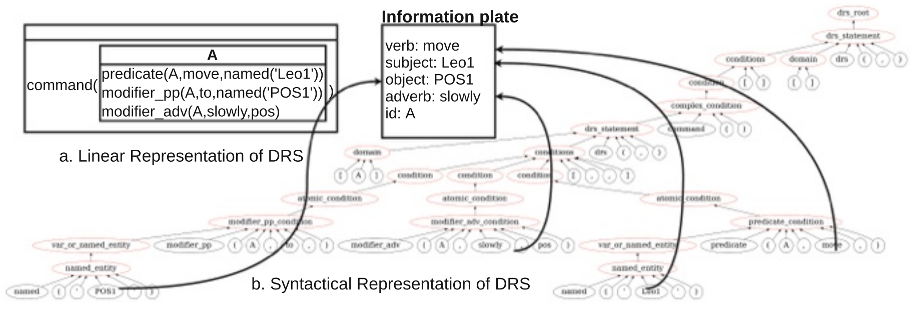

Information plate
verb: move
subject: Leo1
object: POS1
adverb: slowly
id: A
A
command(
predicate(A,move,named('Leo1'))
modifier_pp(A,to,named('POS1'))
modifier_adv(A,slowly,pos)
)
a. Linear Representation of DRS
b. Syntactical Representation of DRS

Fig. 3. Demonstration of semantics extraction from core statements.

2.2 Problem Formulation for Multi-Robot Planning

While single-agent planning problem can be described as classical planning, multi-robot planning requires a temporal model for concurrency and collaboration. Following the temporal reasoning framework [3], our framework accepts temporal constraints in the form of durative actions. Temporal constraints provide the requisite expressiveness to precisely model time, which enables concurrent activities and resource optimization in a multi-robot team.

Multi-agent temporal planning imposes unique challenges compared to single-agent planning, primarily due to resource conflicts. These conflicts, which can occur during planning or execution, arise from limited environmental perception and resource access among multiple robots. To prevent conflicts during planning, mutual awareness among robots is developed, to recognize each other's intentions and resource usage. We introduce the concept of mutual exclusivity (mutex) to describe mutual aware constraints. Events are considered mutually exclusive if they affect or rely on the same state variable assignment. More specifically, two events are considered to be mutex if they satisfy the following constraint

$$\text{mutex}(e_1, e_2) \coloneqq (e_1 \rightarrow \neg e_2) \wedge (e_2 \rightarrow \neg e_1) \tag{1}$$

where e_1, e_2 are propositional events. The mutual exclusive concept is extendable for actions.

A planning problem for multi-robot team can be defined formally as follows

$$\mathcal{I} = \left(\mathcal{S}, s^i, \mathcal{S}^g, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{T}, \mathscr{T}\right) \tag{2}$$

where S is a discrete set of problem states, $s^i \in \mathcal{S}$ is the initial state, $S^g \subseteq S$ is a set of final states. Each action symbol $a \in \mathcal{A}$ corresponds to a primitive robotic motion. The terms $\mathcal{O}, \mathcal{P}$ are sets of object and predicate symbols. Each transition $t_a \in \mathcal{T}$ defines a transition from $s_i \in \mathcal{S}$ to state $s_{i+1} \in \mathcal{S}$ by applying an action $a \in A$. The term $\mathscr{T}$ is a set of temporal interval constraints where each $r_a \in \mathscr{T}$ is attached to an action transition t_a, describing the duration constraints of the action.

Multi-robot planning problem (2) can be described in PDDL [3] using two files: domain file $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{T}, \mathscr{T})$ and problem file $\left(s^i, \mathcal{S}^g\right)$. The domain file is typically predefined by the knowledge experts, which is based on planning scenario and the robots' capability. Additionally, the domain file contains temporal constraints and mutex expressions, enabling concurrent collaborating and task distribution for multi-agent planning. The problem file $\left(s^i, \mathcal{S}^g\right)$ defines the initial state and goal states of the planning problem. CRL-based plan generation algorithm provides a way to automatically construct $\left(s^i, \mathcal{S}^g, \mathcal{T}, \mathscr{T}\right)$ from complex linguistic instructions and temporal specifications.

2.3 Plan Generation Algorithm for Multi-robot Planning

To generate initial and goal states $\left(s^i, \mathcal{S}^g\right)$, the algorithm searches for command-type conditions within tree-DRS. Given a command-type DRS statement

$$\begin{aligned} \mathrm{drs}(a) = \mathrm{drs}([], \mathrm{command}([&X_1, X_2, ..., X_n, Y], \\ &\mathrm{predicate(Y, a, X_1, X_2))))} \end{aligned} \tag{3}$$

the generation algorithm constructs a corresponding goal statement (a X_1 X_2), indicating an event a with two parameters X_1,X_2. For action signatures containing more than two parameters, additional information plates are extracted from other DRS conditions such as "object" and "adverbs". These plates are unified with the base goal statement and construct a complete action statement. The constructed action statements are aggregated and propagated back to the (: `goal`) section.

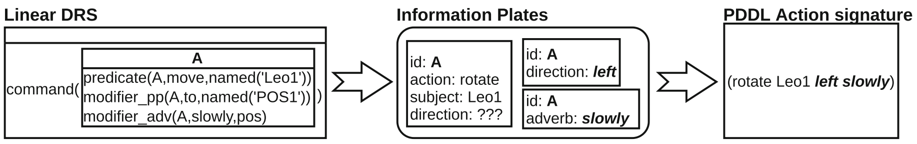


Fig. 4. Multi-parameter action statement construction from command-type DRS condition.

Figure 4 illustrates how an action signature with three arguments (`rotate`) is constructed. From command-type condition, only the action symbol and subject are obtainable. To identify the rotation direction and velocity, the framework searches all DRS adverbial-conditions matching the `rotate` verb ID. Partial information plates are extracted from these conditions, encapsulated by the same ID, and merged with the command-type plate through ID referencing. This process extends to additional arguments, ensuring the complete action statement.

Imperative sentences are commonly preferred for human instructions, but they often lack clarity about the actor in multi-robot team. For instance, the command "Please follow the first robot!" is valid in CRL [8] but not applicable in a multi-agent context. To address this issue, we designate one robot as the default actor to execute target actions when the subject is unspecified.

2.4 Temporal Constructions from Linguistic Descriptions

To generate temporal constraints from linguistic description, the framework focuses on preposition-condition DRS

$$\mathrm{drs}_{\mathrm{pp}}(q) = \mathrm{drs}([X, Y], \mathrm{modifier\_pp}(Y, q, X)) \tag{4}$$

Using Bottom-Up traversal algorithm, we can extract a corresponding temporal information plate $\langle \mathrm{id} : Y, \mathrm{duration} : X_1 \rangle$, where Y is the verb ID. Only preposition DRS statements with temporal lexicon – $q \in \{\mathrm{during}, \mathrm{within}, \mathrm{in}\}$ – are iterated. From temporal information plate, we construct the corresponding temporal constraint $r_Y \in \mathscr{T}$ in PDDL syntax, using (: `duration`) section. The temporal constraint r_Y is updated directly into the domain file. The extracted duration X is converted into seconds. Figure 5 illustrates an example of how a temporal constraint is constructed from linguistic specifications. Due to the limitation of temporal model supported by PDDL, constraints in linear-time temporal logic (LTL) syntax are currently not feasible.

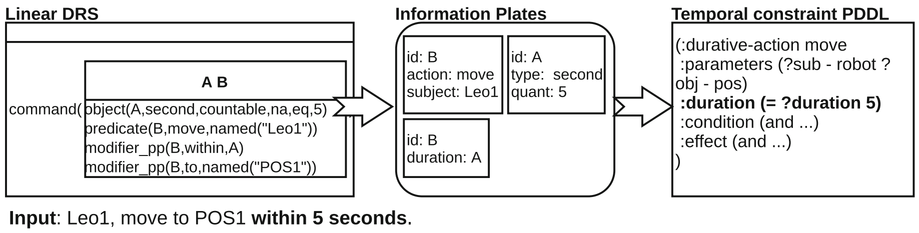
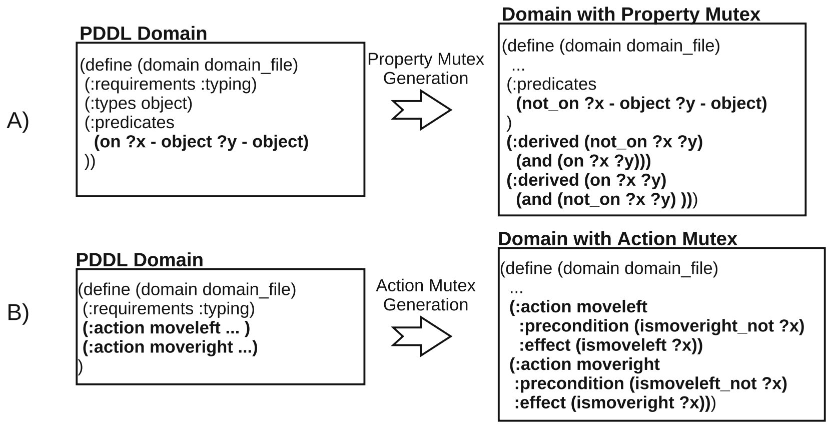


Fig. 5. Temporal constraint construction from linguistic inputs.

2.5 Mutex Synthesis Algorithm

To manage concurrency, mutual exclusive operators are defined. There are two types of mutual exclusive expressions: property mutex and action mutex. Property mutex prevents simultaneous events during the planning phase. Meanwhile, action mutex prohibits incompatible actions during the execution phase, e.g., moving left and right at the same time. Describing mutex expression using CRL is challenging, due to its syntax limitation. Additionally, mutex expressions should be generalized as rules in the domain file, instead of being specified in the problem script. Therefore, instead of converting linguistic descriptions into mutex in PDDL, we developed a synthesis algorithm that automatically generates an updated domain file with mutex constraints. The algorithm handles property and action mutex separately.

To generate property mutex, the algorithm searches for all properties available in the (: `predicates`) section of the domain file. For each predicate $p \in \mathcal{P}$, we construct a new predicate negation $p_{\text{not}} \in \mathcal{P}$. A mutual exclusive relation between p and p_{not} is defined within (: `derived`) section, following (1). An example of property mutex generation is visualized in Fig. 6a. To handle action mutex, the logical relation (1) is defined within (: `precondition`) and (: `effect`) sections (Fig. 6b).



```
(define (domain domain_file)
 (:requirements :typing)
 (:types object)
 (:predicates
   (on ?x - object ?y - object)
 ))
```

```
(define (domain domain_file)
  ...
 (:predicates
   (not_on ?x - object ?y - object)
 )
 (:derived (not_on ?x ?y)
   (and (on ?x ?y)))
 (:derived (on ?x ?y)
   (and (not_on ?x ?y) )))
```

```
(define (domain domain_file)
 (:requirements :typing)
 (:action moveleft ... )
 (:action moveright ...)
)
```

```
(define (domain domain_file)
  ...
  (:action moveleft
    :precondition (ismoveright_not ?x)
    :effect (ismoveleft ?x))
  (:action moveright
   :precondition (ismoveleft_not ?x)
   :effect (ismoveright ?x)))
```

Fig. 6. Property mutex and Action mutex synthesis on the predefined PDDL domain.

After integrating temporal constraints and mutex expressions into the PDDL script, a temporal solver is applied to find suitable temporal plans [5]. In this paper, we utilize Partial Order Planning Forwards (POPF) [2] to find a concurrent solution for multi-robot team. Combining forward state-space search with partial-order planning techniques, POPF can effectively find temporal planning solutions by introducing minimal ordering constraints. POPF uses a Simple Temporal Network (STN) to represent durative action and temporal constraints. As actions are added to the plan, their start and end time points are incorporated into the STN with relevant ordering constraints. The final STN output provides a valid temporal solution that satisfies all temporal constraints. This solution is then dispatched to the multi-robot team, triggering appropriate actions on each robot. Each primitive action is associated with a Behavior Tree, allowing further decomposition into smaller modular actions, which enhances the system's flexibility and adaptability

3 Experiments

We evaluated the proposed method through two main experiments. The first experiment assessed the linguistic characteristics of the framework, aiming to demonstrate that key properties are preserved such as determinism, general domain applicability, and expressiveness from classical CRL [8]. The second experiment evaluated the method's stability of the method on a real-time multi-robot team in navigation missions.

3.1 Linguistic Model Evaluation

To compare this approach with classical CRL, we conducted an evaluation using the natural language instruction dataset [8]. This dataset was specifically developed for robotics and task-planning, comprising 335 planning scenarios and approximately 4,000 tokens. Each entry in the dataset represents a distinct planning scenario, and consists sequence of instructions, queries, and perception descriptions. The linguistic properties of the framework are evaluated on four core tasks: POS tagging, syntactic parsing, semantic parsing, and plan-generation. For POS tagging, we used multi-label accuracy. For syntactic parsing, semantic parsing, and plan-generation, we measured the framework' s feasibility in generating meaningful output, as used in [8].

Among 335 planning scenes, the proposed framework achieves 100% accuracy in POS tagging and syntax success rate, demonstrating the exceptional linguistic robustness and reliability of CRL-based method. Compared to the individual robot framework [9], we observe modest yet significant improvements: semantic parsing accuracy increased from 78% to 81%, while plan-generation success rate rose from 75% to 78%. These improvements can be attributed to the CRL's refined focus on subject identification and resource management in the multi-robot domain, highlighting the framework's adaptability to more complex scenarios.

3.2 Performance Evaluation

We integrated the framework into a real-time multi-robot system of three Leo Rovers. The linguistic framework was incorporated into the robot system via WebSocket communication. For each robot, we implemented five primitive executable actions: `moveto`, `moveforward`, `moveback`, `moveleft`, `moveright`. The `moveto` action requires two arguments (subject, object), while the other only requires one argument (subject). Temporal constraints were explicitly described by participants, while mutex expressions were carefully defined by knowledge experts. To find temporal solutions for the generated plans, we employed the

POPF planner, which efficiently dispatches concurrent actions to the robotic system when a viable solution is identified.

We conducted a study with five participants to evaluate the natural language control of our multi-robot team. Each participant was tasked with describing their desired goals using natural language given the available actions and robot identities within the system. The descriptions are requested to be complex, containing multiple actions, and involving all the robots. The participants were allowed to include temporal constraints. For each set of instructions, we performed 10 trials with varying initial robot states. After each execution, we observed the final state and verified the satisfaction of all the requirements in (:goal) section.

Input: There are 3 robots on the environment. Leo3, move to POS2 slowly. Leo1, move to POS1. Leo2, move to POS3 carefully. Leo3, rotate right after Leo1's first execution. Leo2, rotate left during Leo3's second action.	**Output**: define (problem demo_problem) (:domain demo_domain) (**:objects** pos1 pos2 pos3 - object; ldirect rdirect - direction; leo1 leo2 leo3 - robot; p1 p2 p3 p4 p5 p6 - pid; c - environment; g - execution; i - action; a - robot) (**:init** (after leo3 g) (during leo2 i) (on a c) (first g) (second i) (is_robot leo1) (is_robot leo2) (is_robot leo3)) (**:goal** (and (move leo3 p1 pos2) (move leo1 p2 pos1) (move leo2 p3 pos3) (rotate leo3 p4 rdirect) (rotate leo2 p5 ldirect))))
Input: There are 3 robots on the environment. Leo3, move to POS2 firstly. Leo1, move to POS1 afterward. Leo2, move to POS3 at the same time. Leo3, rotate right after Leo1's first execution. Leo2, rotate left during Leo3's second action. Leo3, move to POS4 after Leo2's second execution. Leo1, move to POS1 before Leo3's last execution.	**Output**: define (problem demo_problem) (:domain demo_domain) (**:objects** pos2 pos1 pos3 pos4 pos1 - object; ldirect rdirect - direction; leo1 leo2 leo3 - robot; p1 p2 p3 p4 p5 p6 p7 p8 - pid; c - environment; g - time; h l n - execution; a - robot; j - action) (**:init** (at leo2 g) (after leo3 h) (during leo2 j) (after leo3 l) (before leo1 n) (on a c) (same g) (first h) (second j) (second l) (last n) (is_robot leo1) (is_robot leo2) (is_robot leo3)) (**:goal** (and (move leo3 p1 pos2) (move leo1 p2 pos1) (move leo2 p3 pos3) (rotate leo3 p4 rdirect) (rotate leo2 p5 ldirect) (move leo3 p6 pos4) (move leo1 p7 pos1))))
Input: There are 3 robots on the environment. Leo1, move to POS2. Leo2, move to POS1 at the same time. Leo3, move to POS3 concurrently. Leo1, rotate left while Leo2's first exectution.	**Output**: define (problem demo_problem) (:domain demo_domain) (**:objects** pos2 pos1 pos3 - object; ldirect rdirect - direction; leo1 leo2 leo3 - robot; p1 p2 p3 p4 p5 - pid; f - time; h - execution; c - environment; a - robot) (**:init** (at leo2 f) (while leo1 h) (on a c) (same f) (first h) (is_robot leo1) (is_robot leo2) (is_robot leo3)) (**:goal** (and (move leo1 p1 pos2) (move leo2 p2 pos1) (move leo3 p3 pos3) (rotate leo1 p4 ldirect))))

Fig. 7. Generated plans with temporal constraints from complex instructions for multi-robot team.

We evaluated the performance of the method on the real-time multi-robot system across various initial states. Each linguistic description contains both subject, object, and temporal expressions indicating the order of executions. Figure 7 showcases examples of these complex linguistic instructions alongside their corresponding generated plans. The outputs are complete and deterministic, and ready for direct interpretation by the POPF solver. These generated plans seamlessly integrate temporal conditions (e.g., `after`, `during`, `before`, `while`), precisely specifying the sequence of actions defined in the original inputs. This demonstrates the method's capability to handle sophisticated, time-sensitive multi-robot coordination tasks.

Input: There are 3 robots on the environment. Leo3, move to POS2 slowly. Leo1, move to POS1. Leo2, move to POS3 carefully. Leo3, rotate right after Leo1's first execution. Leo1, move to POS4 at the same time of Leo3. Leo2, move to POS1 after Leo3's second execution. Leo3, rotate right at the same time. Leo1, rotate left twice at the same time. Leo3, move to POS0, then move to POS3. Leo2, move to POS2 while Leo1 move to POS4.

Input: There are 3 robots on the environment. Leo1, move to POS3. Leo2, move to POS1 concurrently. Leo3, move to POS4 afterward. Leo1, rotate left after the first execution. Leo2 and Leo3, rotate left concurrently. Leo2, move to POS2. Leo1, move to POS4 concurrently. Leo3, move to POS1 at the same time. Leo2, move to POS0 afterward. Leo3, move to POS2 concurrently. Leo1, move to POS1 concurrently.

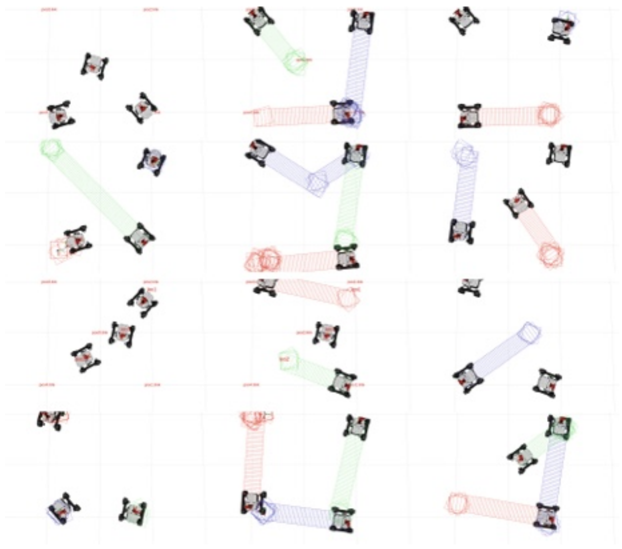


Fig. 8. Plan executions of multi-robot team using the generated temporal plan. The order of executions (left-right, top-down) is consistent with temporal constraints defined in commands.

The experiment demonstrated that multi-robot team consistently interpreted and executed temporal linguistic instructions across all trials, regardless of the initial states. Figure 8 illustrates different executions of the multi-robot at different time steps. The order of robot executions is defined by the temporal solver POPF, after considering all the temporal constraints declared in the generated plans.

4 Conclusion

This paper presents an enhanced Controlled Robot Language (CRL) based communication channel for multi-robot systems. Performance evaluations demonstrate improvements in semantic parsing and plan generation, which confirm the linguistic robustness of the CRL-based approach. Real-world experiments with Leo Rovers in complex planning scenarios further validated the reliability of the communication channel. In conclusion, this research underscores the potential of the CRL-based approach as an effective and user-friendly communication solution for multi-robot domains.

References

1. Beetz, M., et al.: Robosherlock: unstructured information processing for robot perception. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 1549–1556 (2015)

2. Coles, A., Coles, A., Fox, M., Long, D.: Forward-chaining partial-order planning. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 20, pp. 42–49 (2010)
3. Fox, M., Long, D.: Pddl2. 1: an extension to pddl for expressing temporal planning domains. J. Artif. Intell. Res. 61–124 (2003)
4. Liu, R., Guo, Y., Jin, R., Zhang, X.: A review of natural-language-instructed robot execution systems. AI 5, **3**, 948–989 (2024)
5. Mudrova, L., Lacerda, B., Hawes, N.: Partial order temporal plan merging for mobile robot tasks. In: European Conference on Artificial Intelligence (ECAI), pp. 1537–1545. IOS Press (2016)
6. Spencer, D.A., Wang, Y., Humphrey, L.R.: Trust-based human-robot interaction for multi-robot symbolic motion planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1443–1449. IEEE (2016)
7. Tellex, S., Knepper, R., Li, A., Rus, D., Roy, N.: Science and systems, asking for help using inverse semantics. In: Robotics: Science and Systems Foundation (2014)
8. Tran, D., Li, H., He, H.: Ai planning from natural-language instructions for trustworthy human-robot communication. In: International Conference on Social Robotics, pp. 254–265. Springer (2023)
9. Tran, D., Yan, F., Yihun, Y., Tan, J., He, H.: A framework of controlled robot language for reliable human-robot collaboration. In: International Conference on Social Robotics, pp. 339–349. Springer (2021)