AI Planning from Natural-Language Instructions for Trustworthy Human-Robot Communication

Dang Tran, Hui Li, and Hongsheng He(✉)

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

Abstract. Having deterministic communication between humans and robots is essential for a safe, reliable, and trustworthy workspace. Despite the extensive efforts in training robots to comprehend human instructions, the predominant focus has been on improving the generative aspects of models rather than the determinism. This paper presents a frame-based method using planning language and Controlled Robot Language to construct a reliable and deterministic linguistic channel. The model takes multiple instructions as input and generates an appropriate syntactic and formal representation. Core information is extracted from formal representation using bottom-up visitors. The obtained information is used to generate a planning script in Planning Domain Definition Language (PDDL), which can be used directly to control the robot system. The experiment demonstrated great performance of the proposed method on many text-processing tasks and promising results in deterministic communication with robots on a manually created new dataset focusing on the robotic domain.

Keywords: Planning · Natural Language Processing · Human Robot Interaction · Reliable Communication

1 Introduction

In recent years, social robots have become increasingly involved in many aspects of human lives including education [1], healthcare [2], manufacturing [3], and agriculture [4]. Many of these applications require social robots to interact with people on both communicative and physical levels. For instance, healthcare robots can provide patients with medicine information and retrieve the medicine for them [5]. A collaboration robot can engage in dialogue with human partners, assisting them in executing assigned tasks [6]. In these scenarios, it is imperative for the robots to accurately and deterministically comprehend human instructions, thereby mitigating the potential for physical and life-threatening accidents.

This research is supported by NSF #2129113 and NSF #2239540.

A. Al. Ali et al. (Eds.): ICSR 2023, LNAI 14454, pp. 254–265, 2024.
https://doi.org/10.1007/978-981-99-8718-4_22

To develop reliable communication, current methods integrate Natural Language Processing (NLP) with AI Planning. A notable innovation within this field is the automatic generation of Planning Domain Definition Language (PDDL) from natural language inputs. PDDL offers a formal framework for deterministic planning, addressing real-world problems, especially for cognitive robot systems. Various research attempts have been undertaken to address this challenge, such as the dependency-tree-based method [7], the Deep Q-Network method for action templates matching [8], and the dictionary-based approach [9]. The majority of these studies incorporate the model acquisition tools LOCM and LOCM2 [10] for PDDL generation. While these models can handle more generative instructions, their generated PDDL scripts are incomplete and error-prone, necessitating additional manual post-processing to make them readable for high-level planners. Besides, the PDDL scripts generated by acquisition tools are for the general domain which significantly differs from the robotic domain. In the robotic domain, planning scripts should contain only executable, non-abstract actions, with a small number of arguments due to practical constraints.

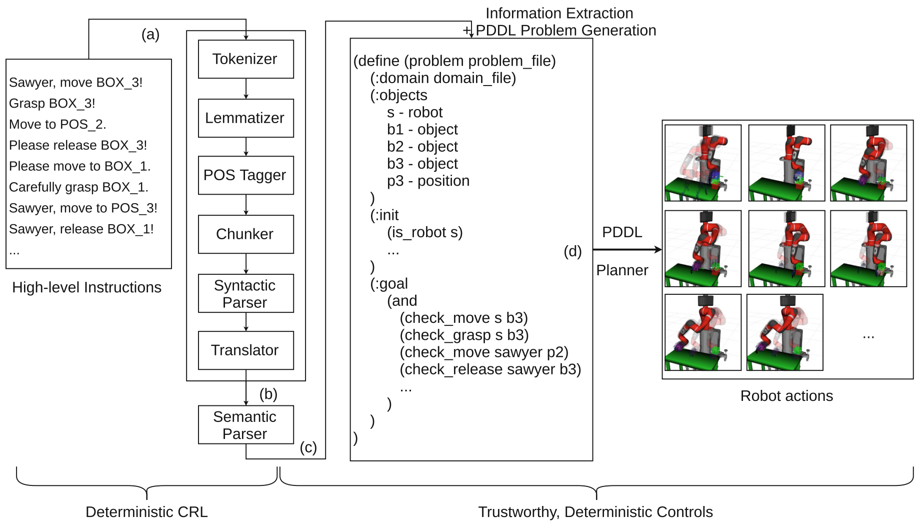
To address these limitations, we present a robust and reliable NLP framework that can produce deterministic robotic planning descriptions. Such determinism can enhance the stability and trustworthiness of human-robot collaborations. The proposed model approaches the generation problem as a multi-stage NLP process, which provides a great control over model performances at different stages. Furthermore, the proposed model focuses on the robot domain, which produces complete PDDL descriptions, that are comprehensible by planners and executable by robot controls. Additionally, the method provides both syntactic and semantic representations, offering a wide range of flexibility and adaptability. The workflow of the proposed method is summarized as follows: Given high-level instructions, the proposed NLP pipeline performs a series of processing tasks on the input, creates syntactic and semantic structures. Subsequently, information is extracted from the semantics, which is then utilized to generate essential PDDL problem sections. The generated PDDL is directly employed for robotic control through ROS-Plan. The contributions of the paper are summarized as follows:

1. A deterministic linguistic communication channel has been implemented. By exploiting the determinism property of AI Planning, we proposed a PDDL-based framework that enables robots to unambiguously understand human instructions.
2. The proposed linguistic dataset is designed for practical robotic planning scenarios, making it not only suitable for linguistic model evaluation but also for demonstrating the practical application of the framework on motion planning.

2 Automated Planning Problem Generation

The proposed linguistic communication method takes natural language instructions from users and generates problem scripts in PDDL, as illustrated in Fig. 1. The raw input is first pushed into the developed linguistic model, named

Controlled Robot Language (CRL) [11]. CRL is a multi-stage linguistic model, where each stage is dedicated to a specific text-processing task. The output of one stage becomes the input for the next stage. In general, given a natural language command, CRL will return an equivalent semantic expression in Discourse Representation Structure (DRS) [12]. This formal structure is then used by various information extraction models to extract essential data for subsequent generation processes. Each information extractor traverses the semantic structure to capture relevant information. The output of the information extractor is represented in a dictionary form, which consists of a list of key-value pairs. Finally, PDDL generation model uses these acquired pairs to create a PDDL problem script. The resulting PDDL problem script is then combined with the developed PDDL domain script that is constructed based on prior knowledge of the planning problem to control the robot.



Sawyer, move BOX_3!
Grasp BOX_3!
Move to POS_2.
Please release BOX_3!
Please move to BOX_1.
Carefully grasp BOX_1.
Sawyer, move to POS_3!
Sawyer, release BOX_1!
...

```
(define (problem problem_file)
   (:domain domain_file)
   (:objects
      s - robot
      b1 - object
      b2 - object
      b3 - object
      p3 - position
   )
   (:init
      (is_robot s)
      ...
   )
   (:goal
      (and
         (check_move s b3)
         (check_grasp s b3)
         (check_move sawyer p2)
         (check_release sawyer b3)
         ...
      )
   )
)
```

Fig. 1. System overview: natural language instructions are translated into Controlled Robot Language (CRL), which returns a CRL-valid expression (a); Semantic compiler produces semantic expression from CRL-valid input (b); Information extraction from semantics to generate a deterministic plan in PDDL (c); The planning script is utilized for robotic control (d).

2.1 Natural Language Disambiguation and Semantics Parsing

To enable the robot to comprehend instructions, we have developed a model that can deterministically derive the formal expression of the input, known as Controlled Robot Language (CRL) [11]. CRL encompasses two primary steps: (1) A processing step is accomplished by a linguistic model, to conduct syntactical analysis, error detection and translation; (2) An operation of the semantic compiler on the translated input, to produce an equivalent and complete semantic expression. The CRL's pipeline comprises six core components: tokenizer, lemmatizer, POS tagger, phrase-chunker, syntactic parser, and translator. The output of this pipeline is a grammatically correct sentence that has been translated from the original input. The translated sentence is directly readable by the semantic compiler. In this paper, we chose Fuchs's discourse-based model [13] as a semantic compiler. While first-order logic is the more common choice for semantic representation, it is only suitable for representing individual, isolated sentences. In more extensive and complex contexts, first-order logic often falls short in capturing relationships between sentences. We leverage discourse-level structure [12] to effectively capture the semantics of the entire robotic planning scene.

In the POS tagging process, we developed a three-layer cascade model with remarkable flexibility and performance control. At the base layer, we employ the spaCy model, a pre-trained statistical and rule-based linguistic model designed for POS tagging. Nonetheless, the performance of spaCy is imperfect, leading to numerous incorrect and inapplicable labels. These inaccuracies can significantly impact the performance of subsequent stages. To flexibly control the model performance, the cascade model introduces two additional layers: dynamic and static layers, which are stacked on top of the base layer. The static layer focuses on correcting inapplicable labels – labels that are correctly identified by the base layer but are unsuitable for the subsequent stages' processing. These inapplicable labels include proper names, keywords, and annotations. In contrast, the dynamic layer is used to rectify incorrect labels caused by limitations of the base layer. Human users can easily detect these mislabeled tokens and update the syntactic rules to the dynamic layer. Given a tokenized instruction, the dynamic, static, and base layers carry out the tagging process respectively. The final outcome is a sequence of POS tags, cascading from the top layer to the bottom. If using only the base layer, the model may inaccurately label certain tokens. For instance, "move" can be labeled as a singular noun (NN) rather than its correct base form verb (VB). The dynamic layer rectifies this by ensuring that "move" can only be VB. Although the base layer can still correctly identify the POS of many tokens, such as exclamation and commas, the resulting non-alphabetical labels are not comprehensible to the syntactic and semantic parsers. Hence, the static layer is used to assign more suitable labels to these tokens.

To increase the grammatical expressiveness of the proposed model, we developed a translator capable of converting invalid sentences to valid ones. Given the inherent error-proneness of natural language commands, users may inadvertently provide invalid inputs. The proposed translator can detect, navigate and fix the errors using their syntactic structures. To capture the syntactic structure of a sentence, we employed a Context Free Grammar (CFG) specifically designed for Controlled Robot Language. This grammar consists of 220 meticulously selected grammar rules. The lexicons are dynamically generated using the inputs and their POS labels. Our dynamic CFG grammar is generative enough to capture the syntactic structure of both invalid and valid sentences. If a user enters an invalid command, the syntactic parser can still construct the sentence's syntactic structure. Leveraging this structure, the translator can identify the exact locations where the errors have occurred and translate them into the correct forms. Figure 2 represents an example of how a translation is accomplished. The dynamic CFG is robust enough to capture the typos (verb "moves" should be in second person form). Once the tree is captured, the model can detect the precise location of the error, which is under the nonterminal Verb. The verb is then transformed into the correct form, resulting in a valid, grammatically correct sentence.

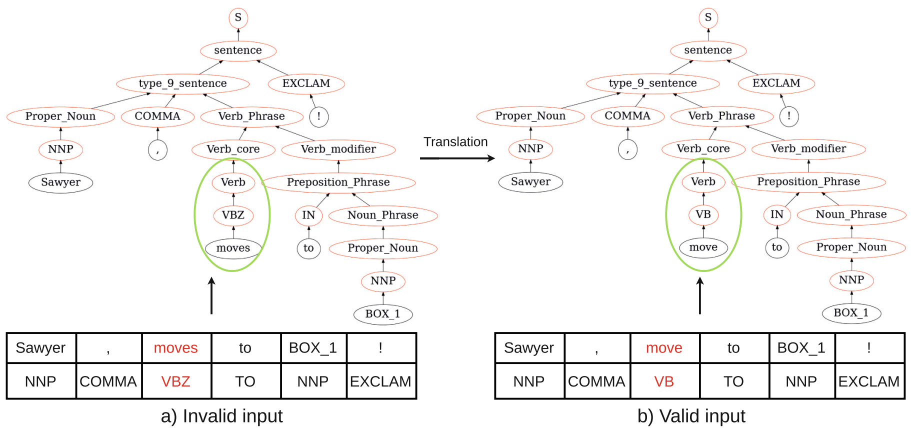


Fig. 2. Comparison of syntactical trees during translation from invalid to valid sentence. Translation is used to detect and repair the misuse of the verb tense "moves".

2.2 Identification of PDDL Problem Components

Once all instructions are CRL-valid, we construct the semantic representation of the entire planning scene. To effectively represent large and complex contexts, we utilize DRS [12] as a semantic compiler, which can capture linked anaphora and coherence. In the complete semantic expression, we focus more on key components, including verbs, object roles, functions, and properties, which are essential for the subsequent generation process. Figure 3 illustrates how verb, subject and object are extracted from semantic statement. The box-like DRS formulation (Fig. 3a) is first linearized into an equivalent DRS statement, which is further parsed into a tree-like structure (Fig. 3b). Given this parsed semantic tree, we can easily extract and capture the essential information. A recursive algorithm traverses along the tree, detecting and copying the matched information into an output dictionary, which contains a set of key-value pairs.

2.3 Plan Script Generation

To deterministically generate a planning script, we developed a PDDL generation model. The PDDL generator is a composition of various visitors, each designed

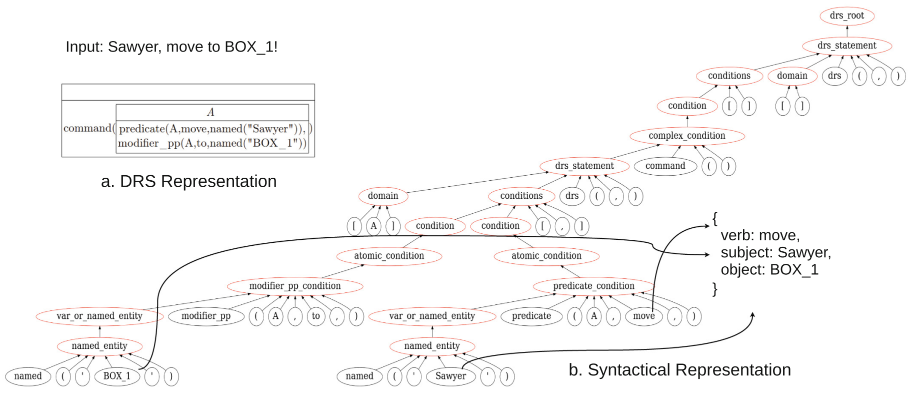


Fig. 3. Two different but equivalent representations of the formal expression. Tree-like format of the formal expression is easier for information extraction and PDDL generating.

to gather specific sets of information. Although the logic flows of these visitors can overlap, the complexity of the task necessitates the use of multiple visitors. In practice, every PDDL generator requires a minimum of 3 distinct visitors: 1) A predicate-visitor to capture executable verbs and function arguments; 2) An object-visitor to capture proper nouns and identifying entities; 3) A preposition-visitor to capture geometric information. The captured outputs are combined together to generate PDDL problem sections. The PDDL problem file requires 3 main sections: (**:objects**), (**:init**), and (**:goal**). The generated sections are combined together into one problem description script, which is readable by PDDL planners. The generation strategies depend strongly on the type of instruction.

The generated PDDL problem file alone is insufficient for planning and robotic control. PDDL planner requires both problem and domain files. Since human instructions do not provide useful information to construct a PDDL domain, we create a domain file based on the prior knowledge of the planning domain. For instance, in the Block world problem, we defined a set of actions that are most suitable and executable in the robot framework: *move*, *grasp*, *release, throw,* and *search*. Following the design conventions for robotic work cells [14], we have also defined various types of objects that are relevant for the manipulation environment: *robot*, *position*, *object* and *pid* (process id). Figure 4 represents a complete PDDL domain for the Block world problem.

The diversity of semantic expressions makes it challenging to develop a universal generation algorithm. Therefore, it becomes necessary to implement various generation strategies tailored to the specific types of instructions. In practice, we have categorized all instructions into 3 primary types: 1) Single action sentence with well-identified entities; 2) Single action sentence with ungrounded entities; and 3) Sentence involving multiple actions.

Type 1 commands consist of sentences featuring a single action and clearly identified entities. The plan generation strategy for type 1 commands is straight-

```
(define (domain demo_file)
   (:requirements :typing)
   (:types
      position_and_object
      table robot
      object - position_and_object
      position - position_and_object
      process_id
   )
   (:predicates
      (is_robot ?x - robot)
      (at ?x - object ?y - position)
      (on ?x - object ?y - position_and_object)
      (check_move
         ?robot - robot
         ?pid - process_id
         ?x - position_and_object)
      (check_grasp
         ?robot - robot
         ?pid - process_id
         ?y - object)
      (check_release
         ?robot - robot
         ?pid - process_id
         ?y - object)
    )
```

```
   (:action move
      :parameters (?robot - robot
         ?pid - process_id
         ?y - position_and_object)
      :precondition (and)
      :effect (check_move
         ?robot ?pid ?y)
   )
   (:action grasp
      :parameters (?robot - robot
         ?pid - process_id
         ?y - object)
      :precondition (and)
      :effect (check_grasp
         ?robot ?pid ?y)
   )
   (:action release
      :parameters (?robot - robot
         ?pid - process_id
         ?y - object)
      :precondition (and)
      :effect (check_release
         ?robot ?pid ?y)
   )
)
```

Fig. 4. Develop PDDL domain for Block world problem.

forward and requires no additional processing steps. The dictionary produced by predicate-visitor serves as the basis for the construction of (**:goal**) section. Type 1 commands are the simplest instructions involving well-defined entities. To convert a sentence into a type 1 equivalent, we replace all available nouns with recognized identities. Figure 5a illustrates an example of how a type 1 command is created, wherein the unidentified noun (*the red box*) is replaced with an identified proper noun (BOX_1).

Type 2 commands represent the relaxed version of type 1 commands, allowing ungrounded objects. In other words, type 2 commands can include singular or plural nouns. To generate plans for type 2 commands, an additional preprocessing step is required, which involves mapping unidentified nouns to objects within the robot's perception. This process is referred to as *grounding*, where we establish a correspondence between objects in the natural language instructions and the objects available in the robot's knowledge base. Figure 5b illustrates an example of how the proposed model handles type 2 commands.

Finally, type 3 commands contain more than one action, which is an alternative formulation for multiple sentences. Since each predicate-visitor is optimized to handle one primary action and its associated arguments, sentences with multiple actions can introduce complexity to the extraction process. Therefore, a distinct strategy is needed to generate plans for type 3 instructions. To handle type 3 commands, we convert each command into multiple sentences with

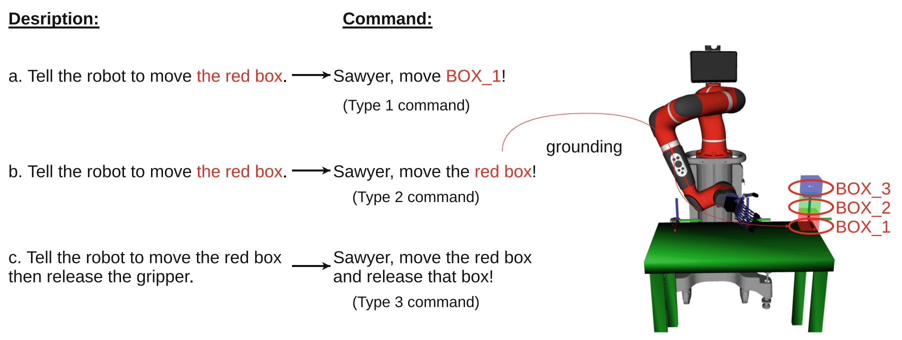

Desription:
Command:
a. Tell the robot to move the red box.
Sawyer, move BOX_1!
(Type 1 command)
grounding
b. Tell the robot to move the red box.
Sawyer, move the red box!
(Type 2 command)
c. Tell the robot to move the red box then release the gripper.
Sawyer, move the red box and release that box!
(Type 3 command)
BOX_3
BOX_2
BOX_1

Fig. 5. Three types of commands that the proposed system can handle efficiently with different generating strategies.

consistent subjects and the same action order. The primary actions and their respective arguments are segmented into distinct blocks.

3 Experiment

To evaluate the effectiveness of the proposed method, we conducted experiments on various real-world planning scenarios. Recognizing the absence of a suitable dataset for evaluation, particularly since most NLP datasets are centered on the broader English domain, we developed a new dataset specifically tailored for robotics and planning. Each entry in the dataset represents a distinct planning scenario, and consists sequence of instructions, queries, and perception descriptions. The performance of the proposed method is measured by the system's competence in comprehending instructions and its proficiency in generating valid plans. More specifically, we evaluate the model's accuracy in four core tasks: POS tagging, syntactic parsing, semantic parsing, and PDDL problem generation. The experiment results demonstrate the robust performance and configurability of the POS tagger and syntactic parser. While there is a lower accuracy performance observed in semantic parsing and PDDL generation. Finally, we illustrate how the generated planning file can be used for controlling a robot in simulation.

3.1 Experiment Setup

To set up the robot environment, we implemented the MagicHand platform [15], with a specific focus on the Block world problem. This platform comprises a Rethink Sawyer robot and an AR10 gripper, both implemented in simulation using Gazebo and ROS Kinetic. The NLP pipeline is developed in Python 3.9, utilizing libraries such as Spacy, NLTK and Lark. For high-level robotic control and PDDL solving, we use Moveit! [16] and Fast-Forward planner [17]. The choice of Fast-Forward planner is significant, since it preserves the order of actions in the planning solution.

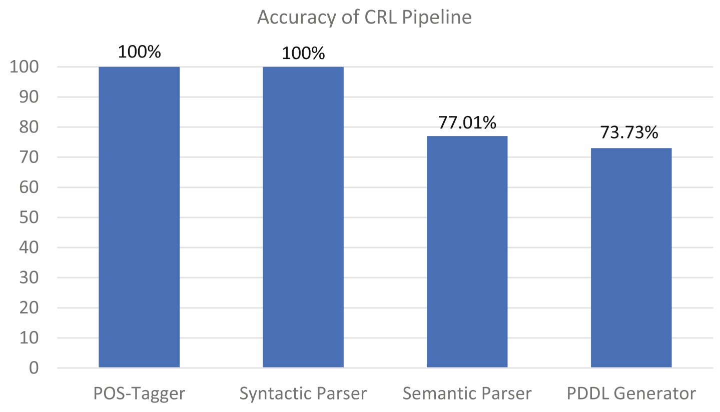


Fig. 6. Performance of the CRL model's components.

The dataset contains 335 planning scenarios, with approximately 4,000 tokens. Each entry represents a sequence of sentences, and these sentences can take the form of a query, descriptive statement, or command. The dataset also includes ungrammatical sentences. We evaluated the model's performance on the following tasks: POS tagging, syntactic parsing, semantic parsing, and PDDL problem generation. For POS tagging, we measured the accuracy of the multi-label model. For more complex tasks like syntactic parsing, semantic parsing, and PDDL generation, we measured the framework's feasibility in generating meaningful output.

3.2 Performance of CRL Model

The performance of CRL model is visualized in Fig. 6. The proposed framework excels with 100% accuracy in POS tagging. Additionally, the model achieves a 100% feasibility score in syntactic parsing, demonstrating its capability to parse all the instructions in the dataset. However, the feasibility score for semantic parser is slightly lower at 77.01% (258/355 sentences). This is attributed to certain limitations of Fuchs's semantic compiler [13]. For instance, Fuchs's model can not understand gerunds, personal pronouns, and adheres to specific rules for adverbs and prepositions. The performance of PDDL planning generation stands at 73.73% in terms of feasibility (247/355 sentences). It's noteworthy that the performance of earlier tasks will set the upper limit for the performance of subsequent tasks. This emphasizes the importance of performance control and model flexibility, which the proposed method addresses. Finally, we showed how one scenario successfully led to the planning execution in the robotic system. The scenario contains 12 commands executed in sequential order. Figure 7 illustrates how the robotic system processes the input and executes the actions accordingly.

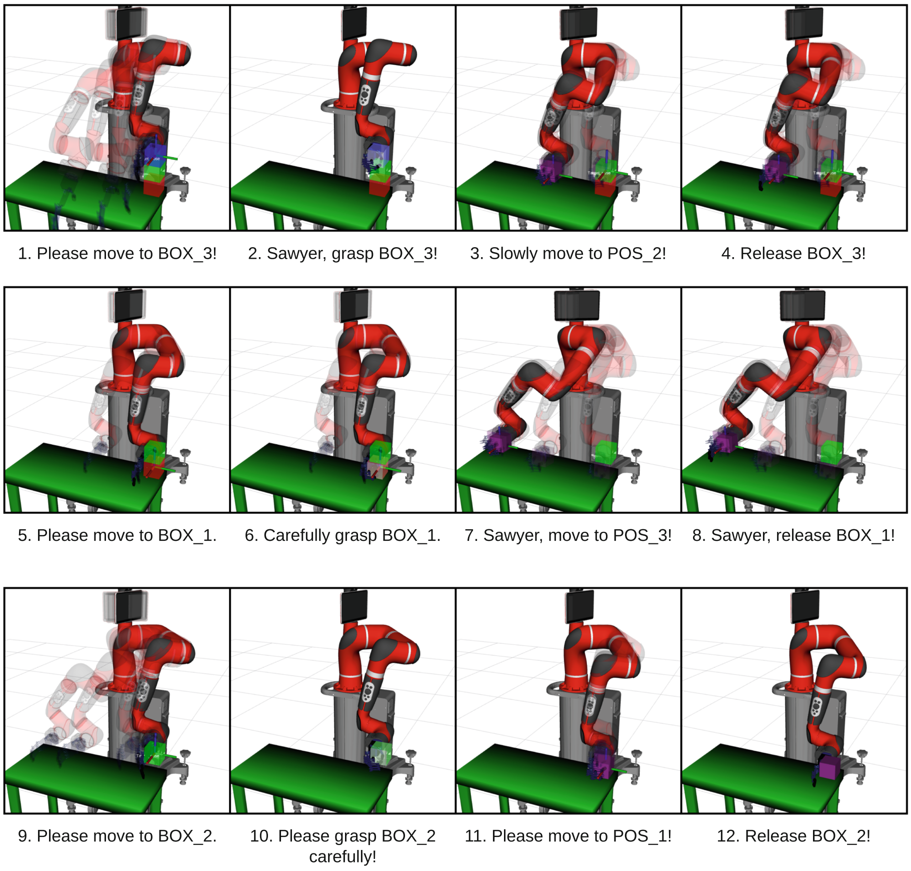


Fig. 7. The proposed system successfully converted a 12 consecutive commands into a practical execution plan.

4 Conclusion

This paper presents a robust and deterministic linguistic communication channel, which allows human users to interact reliably with robots. Leveraging the determinism property of planning descriptions (PDDL), we developed a multi-stage model that can provide both syntactic and semantic expressions. Additionally, the model demonstrates the ability to comprehend instructions and generate deterministic planning. The performance of the method was evaluated on a newly designed dataset, yielding impressive results. POS tagging achieved a perfect 100% score in accuracy, while syntactic parsing secured a flawless 100% feasibility score. These outcomes affirm the deterministic nature of the proposed method. Although the model achieved slightly lower feasibility scores 77.01% for semantic parsing and 73.73% for PDDL problem generation, these limitations can be attributed to the constraints of the semantic compiler itself. In summary, this paper highlights the promising potential of PDDL-based methods for building deterministic communication channels between humans and robots, which can reduce physical accidents in workplace environments.

References

1. Belpaeme, T., Kennedy, J., Ramachandran, A., Scassellati, B., Tanaka, F.: Social robots for education: a review. Sci. Robot. **3**(21), eaat5954 (2018)
2. Breazeal, C.: Social robots for health applications. In: 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 5368–5371. IEEE (2011)
3. Jahanmahin, R., Masoud, S., Rickli, J., Djuric, A.: Human-robot interactions in manufacturing: a survey of human behavior modeling. Robot. Comput.-Integr. Manuf. **78**, 102404 (2022)
4. Sparrow, R., Howard, M.: Robots in agriculture: prospects, impacts, ethics, and policy. Precis. Agric. **22**, 818–833 (2021)
5. Kaiser, M.S., Al Mamun, S., Mahmud, M., Tania, M.H.: Healthcare robots to combat COVID-19. In: Santosh, K.C., Joshi, A. (eds.) COVID-19: Prediction, Decision-Making, and its Impacts. LNDECT, vol. 60, pp. 83–97. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-9682-7_10
6. Lopes, L.S., Teixeira, A.: Human-robot interaction through spoken language dialogue. In: Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113), vol. 1, pp. 528–534. IEEE (2000)
7. Lindsay, A., Read, J., Ferreira, J., Hayton, T., Porteous, J., Gregory, P.: Framer: planning models from natural language action descriptions. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 27, pp. 434–442 (2017)
8. Miglani, S., Yorke-Smith, N.: NLtoPDDL: one-shot learning of PDDL models from natural language process manuals. In: Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS). ICAPS (2020)
9. Yan, F., Tran, D.M., He, H.: Robotic understanding of object semantics by referringto a dictionary. Int. J. Soc. Robot. **12**, 1251–1263 (2020)
10. Cresswell, S., McCluskey, T., West, M.: Acquisition of object-centred domain models from planning examples. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 19, pp. 338–341 (2009)
11. Tran, D., Yan, F., Yihun, Y., Tan, J., He, H.: A framework of controlled robot language for reliable human-robot collaboration. In: Li, H., et al. (eds.) ICSR 2021. LNCS (LNAI), vol. 13086, pp. 339–349. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90525-5_29
12. Kamp, H., Van Genabith, J., Reyle, U.: Discourse representation theory. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic. Handbook of Philosophical Logic, vol. 15, pp. 125–394. Springer, Dordrecht (2011). https://doi.org/10.1007/978-94-007-0485-5_3
13. Fuchs, N.E., Schwitter, R.: Attempto controlled English (ace) (1996). arXiv preprint cmp-lg/9603003
14. Ochi, K., Fukunaga, A., Kondo, C., Maeda, M., Hasegawa, F., Kawano, Y.: A steady-state model for automated sequence generation in a robotic assembly system. SPARK 2013 (2013)

15. Li, H., Tan, J., He, H.: MagicHand: context-aware dexterous grasping using an anthropomorphic robotic hand. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 9895–9901. IEEE (2020)
16. Chitta, S., Sucan, I., Cousins, S.: Moveit! IEEE Robot. Autom. Mag. **19**(1), 18–19 (2012)
17. Hoffmann, J.: FF: the fast-forward planning system. AI Mag. **22**(3), 57 (2001)