2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

PlanCollabNL: Leveraging Large Language Models for Adaptive Plan Generation in Human-Robot Collaboration

Silvia Izquierdo-Badiola[1,3], Gerard Canal[2], Carlos Rizzo[1] and Guillem Alenyà[3]

***Abstract*— *"Hey, robot. Let's tidy up the kitchen. By the way, I have back pain today"*. How can a robotic system devise a shared plan with an appropriate task allocation from this abstract goal and agent condition? Classical AI task planning has been explored for this purpose, but it involves a tedious definition of an inflexible planning problem. Large Language Models (LLMs) have shown promising generalisation capabilities in robotics decision-making through knowledge extraction from Natural Language (NL). However, the translation of NL information into constrained robotics domains remains a challenge. In this paper, we use LLMs as translators between NL information and a structured AI task planning problem, targeting human-robot collaborative plans. The LLM generates information that is encoded in the planning problem, including specific subgoals derived from an NL abstract goal, as well as recommendations for subgoal allocation based on NL agent conditions. The framework, PlanCollabNL, is evaluated for a number of goals and agent conditions, and the results show that correct and executable plans are found in most cases. With this framework, we intend to add flexibility and generalisation to HRC plan generation, eliminating the need for a manual and laborious definition of restricted planning problems and agent models.**

I. Introduction

Let us consider a scenario in which a human initiates a collaboration by saying *"Let's tidy up the kitchen. By the way, I have back pain today"*. In this situation, human agents would be able to use common sense to devise a plan with specific subgoals, such as storing a spoon in a drawer, and to allocate these subgoals appropriately, for example, not assigning the task of tidying up a heavy casserole to the human with back pain. Now, let us consider a Human-Robot Collaboration (HRC) where a similar goal is shared, and a number of specific and executable tasks must be planned and assigned to each agent. In this case, how can a robotic system leverage the available abstract information in a natural and flexible manner to generate an **executable HRC plan** with **specific subgoals** and an **appropriate task allocation**?

AI task planning may be used [1], [2], but has a main drawback: the planning problem, including a world model (domain) with an initial state and goal (problem), needs to be carefully designed. This is a time-consuming task, which often results in a rigid problem definition, difficult to generalise and adapt to a range of different situations.

Alternatively, Large Language Models (LLMs) can generate plans with no need for careful design [3], [4], by extracting common-sense knowledge from Natural Language (NL), such as an abstract goal. However, this domain-independent method suffers from grounding problems, often resulting in plans that are not executable by the robot.

An added challenge arises when planning for HRC, consisting of allocating tasks to different agents based on their strengths, limitations, or preferences. This remains an open question, as agent models are often limited and unrealistic, unable to cover all elements representing the agent states.

In this paper, we propose to combine the strengths of AI task planning and LLMs for effortless and adaptive HRC plan generation. We present PlanCollabNL (see Fig. 1), a framework to generate grounded collaborative plans from three pieces of information: an abstract NL goal, the current scene state specified in NL (objects, locations, agent conditions), and the available tasks defined in the planning domain. The key contributions of this work are:

1) A method leveraging LLMs to derive grounded planning subgoals from an NL abstract goal and the current environment state. This eliminates the need for a manual definition of a specific planning problem goal.
2) A method that uses LLMs to reason about NL agent conditions and influence the task allocation in the HRC plan. This eliminates the need for restricted and unrealistic agent models, and enables a high degree of adaptability to the collaborating agents' conditions.
3) A framework that integrates these methods and translates their acquired common-sense knowledge into a structured task planning problem implemented using Planning Domain Definition Language (PDDL) to efficiently generate grounded plans.

The framework is evaluated for a number of NL abstract goals and agent conditions, with results showing that executable plans with a coherent task allocation between the agents are obtained in most cases.

II. Related Work

AI task planning for HRC. AI (or automated) task planning frameworks have been used to efficiently generate and distribute the sequence of actions required to achieve a shared goal [1], [2], [5]–[8]. Defining the planning problem constitutes a laborious task, and is usually based on restricted,

[1]Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Barcelona, Spain
[2]Department of Informatics, King's College London, United Kingdom
[3]Institut de Robòtica i Informàtica Industrial, CSIC-UPC Barcelona Spain
*S. Izquierdo is a fellow of Eurecat's *Vicente López* PhD grant program. G. Canal has been supported by the Royal Academy of Engineering and the Office of the Chief Science Adviser for National Security under the UK Intelligence Community Postdoctoral Research Fellowship programme. This work was partially financed by MCIN/ AEI /10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR" under the project ROB-IN (PLEC2021-007859); and by the project COHERENT funded by MCIN/ AEI /10.13039/501100011033, Spain, the "European Union NextGenerationEU/PRTR", Spain, (PCI2020-120718-2), and by UKRI (EP/V062506/1).
Correspondence: silvia.izquierdo@eurecat.org

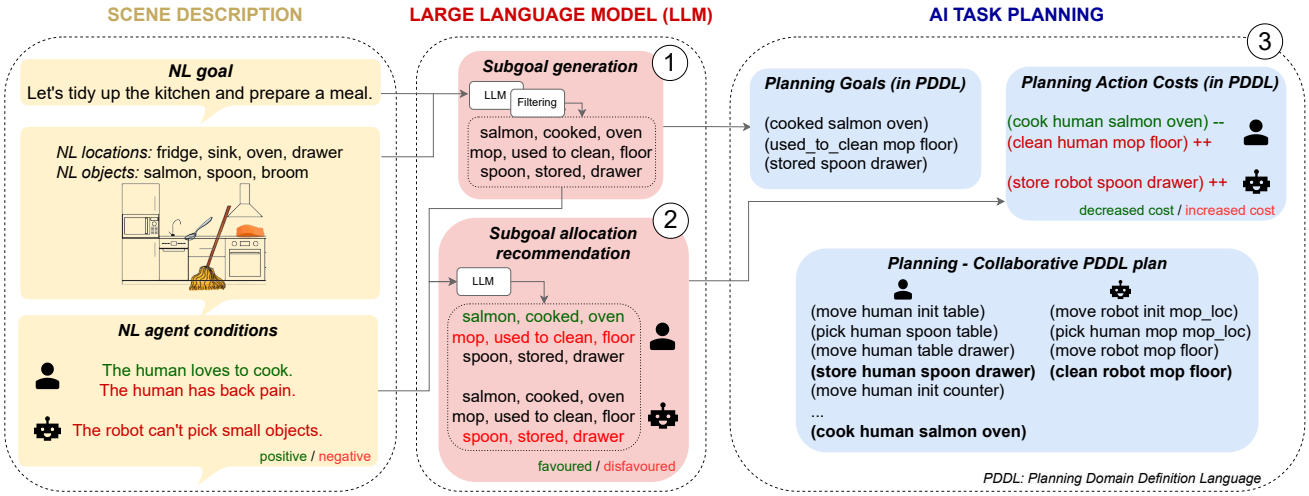

Fig. 1. PlanCollabNL: an LLM is used to generate and filter subgoals from an abstract goal and the current environment (1), and later to reason about the subgoal allocation among the agents, favouring or disfavouring subgoals for each agent based on their conditions (2). This information is encoded into the PDDL planning problem, translating the LLM output subgoals into PDDL goals, and modifying the action costs based on the LLM recommendations. The complete planning problem is given to a planner to produce grounded and executable collaborative plans that consider the agent states (3).

not generalisable and closed-world models. This limits the ability of the systems to deal with complex and abstract tasks, and to adapt the plan to both the environment and the contributing agents. We are thus motivated to take advantage of LLMs' reasoning abilities, whilst maintaining the use of AI planning to guarantee sound, complete and executable solutions, something which LLMs still fail to do [9]. In this work, the planning problem is defined using PDDL [10], a standardised planning encoding consisting of two files: a domain and a problem. The domain includes the definition of the world and underlying rules, whilst the problem contains the initial state and goal conditions. Planners are then able to find a plan based on a search guided by heuristics extracted automatically from the problem representation.

Agent modelling for plan adaptability. For successful HRC, the planner should adapt the plan and assign the appropriate complementary actions to the team members based on their state and capabilities. In previous work [1], we targeted the gap between human modelling and its effective integration in the planning framework. This is something other works have attempted to do [2], [11], with some of the elements modelled including factors such as knowledge [7], [8], capacity, distraction and fatigue [12]–[14]. In these works, the models are fixed and unable to encompass the full complexity of the agents' states. LLMs have been adopted to model and simulate human behaviour [15], [16], or to learn human preferences from past experiences [17]. However, none of the works exploit LLMs to reason about other agents' conditions for collaborative plan adaptation. In our work, we remove the need for a limited agent model, by leveraging LLMs reasoning capacities to influence the plan based on any agent condition expressed in NL form.

LLMs in robotics planning. Recent studies have showcased LLMs' reasoning and few-shot generalisation abilities, motivating their integration into embodied systems for decision-making [18]–[20]. Nevertheless, a major challenge remains in grounding language to the robot capabilities, and dealing with LLMs' hallucinations. We identify two lines of research, where LLMs are used directly as a planner, or as an auxiliary helper bringing knowledge to the planning system. When using LLMs as planners, several methods have been proposed to mitigate grounding issues, including affordance functions for action feasibility [3], or semantic translation of plan steps to admissible actions through an LLM [4]. Other works [21]–[24] add sources of NL feedback such as scene information or precondition errors during plan execution to enable dynamic replanning. In [25], the plan is nested within the LLM-generated policy code for direct execution by the robot. Although these approaches are promising, executability issues are still present, raising the question of whether LLMs are ready to plan and to fully take control of robot behaviours [9]. The second line of research investigates LLMs as an auxiliary tool for planning, noting in [26] and [27] that while LLMs fail to directly solve many problems, they can be useful in guiding the search. They call for further research on the topic and highlight real-time implementation challenges. Other works use LLMs to translate natural language to structured PDDL format, focusing on goals [28], action definitions [29], or full planning problems [30]. Despite achieving promising results, directly generating structured planning language increases the likelihood of syntax errors and unsolvable plans. We build on the idea of combining the strengths of grounded AI task planning with flexible LLM capabilities, bringing adaptability and facilitating the tedious task of defining a rigid planning problem. In this work, the planning domain capabilities are not modified. LLMs reason about planning subgoals, which are then filtered and grounded before being integrated into the planning problem, avoiding executability issues and guaranteeing sound plans. We also notice that none of the works integrating LLMs with task planning address HRC plans, using LLMs to reason about the environment's and agents' states. The uses of LLMs to generate grounded and filtered planning subgoals from an NL goal and to reason about NL agent abilities in

an HRC planning scenario are both novel contributions to the existing literature.

III. Methodology

A. Framework

The implemented framework (as seen in Fig. 1) generates an adapted HRC plan to reach a given abstract NL goal, taking into account the current scene (objects, locations, and agent conditions) and the available actions defined in the planning domain. Regarding the LLM, we utilise the pretrained GPT-3 [31] without further fine-tuning, and apply n-shot learning, where the prompt includes n previous examples. The process consists of three main phases:

1) **Subgoal generation and filtering** (Sec. III-B): The LLM initially divides the abstract goal of the plan (e.g. prepare a meal) into a number of subgoals, in the form of: "object", "task (predicate)", "location" (e.g., "salmon, cooked, hob"), considering the current scene and available actions. These subgoals are generated, grounded and filtered in three stages.
2) **Subgoal allocation recommendation** (Sec. III-C): The LLM reasons about the preferred subgoals' allocation among the agents based on their conditions.
3) **PDDL task planning** (Sec. III-D): The outputs from 1) and 2) are integrated into the PDDL planning problem. The subgoals are parsed and formatted into PDDL goals, and the subgoal allocation recommendation is translated into action costs associated to the relevant subgoals. The final HRC plan can then be generated by a planner from the complete domain and the problem.

B. Generation of Grounded Planning Subgoals from an Abstract NL Goal

The first component of the framework receives an NL abstract goal from the user and generates grounded planning subgoals in the form of "object, predicate, location" (e.g., "spoon, stored, drawer"). The process is done in three stages, depicted in Fig. 2. In the first stage, the system generates a prompt based on a template, incorporating the user's goal and the environment information, and queries the LLM to output a list of subgoals (Fig. 2-Stage 1). The prompt includes five previous examples, made of the goal, available predicates, objects, locations, and the subgoals expected as output. After prompting the LLM with a new goal, the returned list of subgoals is grounded to the available objects, locations and predicates in the planning domain (Fig. 2-Stage 2). Finally, the subgoals are filtered with further templated LLM prompts, generated by the system based on the current subgoals. Incorrect subgoals are filtered out by the LLM based on common sense and on their contribution to the high-level goal (Fig. 2-Stage 3). The full prompts containing the examples are made available[1]. This method provides a robust, natural and effortless way of defining specific planning subgoals for an NL abstract goal, where all stages are essential to ensure the executability of the final plan.

[1]See `http://www.iri.upc.edu/groups/perception/#HRC_TaskPlanningLLM`

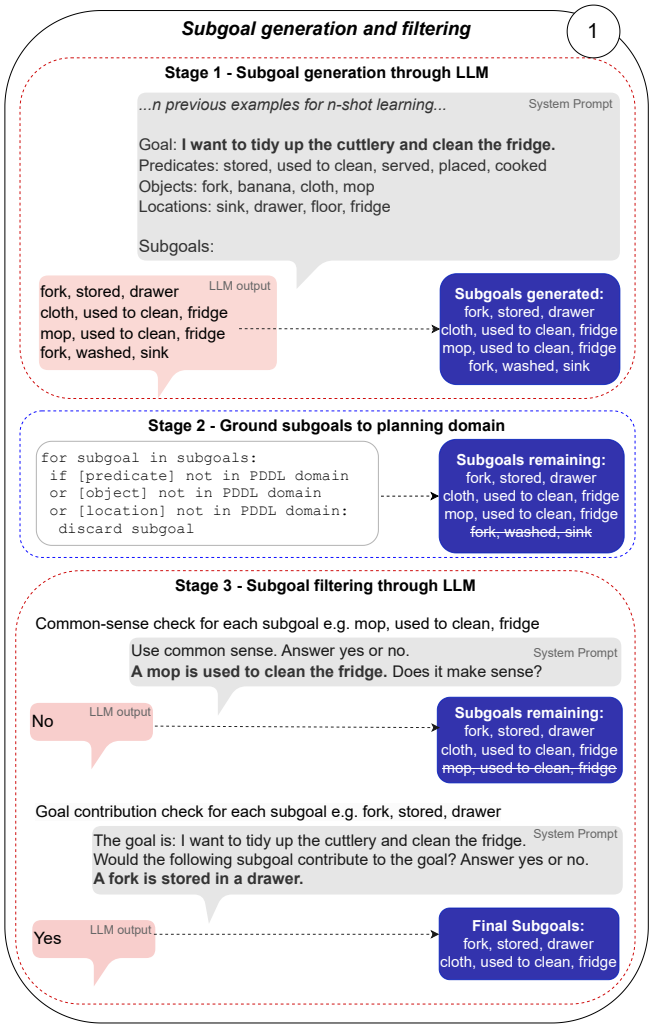

Fig. 2. Subgoal generation stages: in the first stage, an LLM generates a set of subgoals from a prompt constructed by the framework from an NL abstract goal. These subgoals are then grounded to the planning domain capabilities. The third stage filters the remaining subgoals by querying an LLM on whether they make sense and contribute to the high-level goal.

C. Reasoning About Subgoal Allocation Based on NL Agent Conditions

In a collaboration, actions should be assigned to different agents based on their capabilities and preferences. The second component of the framework takes in the filtered subgoals along with the given NL agent conditions, and queries the LLM to identify which subgoals should be favoured or disfavoured for each agent (see Fig. 3). Note that the LLM decision does not force the planner to allocate the actions, but influences the planner allocation, penalising the corresponding actions with a cost. A detailed explanation of this process is provided in the next Section III-D. This approach offers a flexible and efficient means of adapting the plan to any agent condition expressed in NL form, eliminating the need for complex and restrictive agent models.

D. Generalisable PDDL Task Planning Problem Definition

The planning problem is defined in PDDL [32] and consists of a domain, including the object types, predicates,

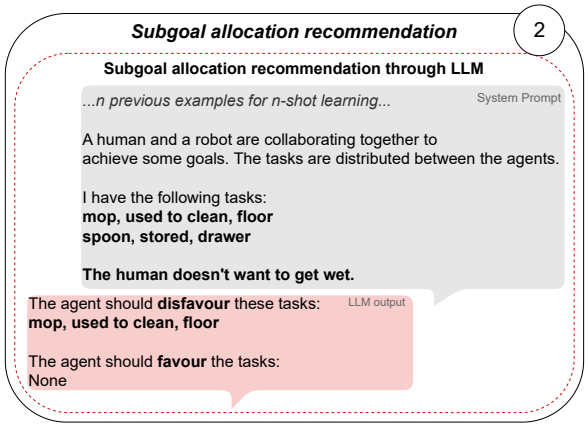

Fig. 3. Subgoal allocation recommendation: The LLM suggests to favour or disfavour certain subgoals for an agent based on their conditions.

functions and actions that can exist within the model, and a problem, collecting what objects exist, what the states of the predicates and functions are, and what the end goal is. The PDDL files are provided at the link in footnote 1.

PDDL domain. In our framework, the domain is fixed and encompasses all agents' capabilities and scenario constraints. It is important to note that the LLM only modifies the PDDL problem and not the domain, ensuring the generation of grounded plans and reducing executability issues. The environment is defined in terms of agents, objects and locations, represented as object types in the domain: *agent, obj, loc*. With the aim of creating a planning definition that is easily extendable to different tasks, we have defined two types of actions and predicates: *standard* and *subgoal* (see Fig. 4). A *subgoal* action (e.g., *store*) has a *subgoal* predicate (e.g., *stored*) as an effect, directly achieving a subgoal which might be part of the goal. It has a linked cost (e.g., *stored_cost*) as an additional effect, associated to the action parameters (*agent, obj, loc*), which increases the total cost of the plan. The *standard* predicates and actions are needed as intermediate steps in the plan to reach the *subgoal* elements, and remain internal to the planning system. A *standard* action (e.g., *move*) has *standard* predicates (e.g. *at_loc*) as effects, and has no cost. This provides a generalisable framework, extendable to a great number of scenarios where an LLM can reason to provide external planning knowledge, as described in the paragraph below. To define a new scenario with new capabilities, the possible tasks must be identified (e.g., painting) and associated to a *subgoal* action (e.g., *paint*), predicate (e.g., *painted*), and cost (e.g., *painted_cost*). This will enable the LLM system to automatically specify subgoals involving the task, and to influence their allocation, as detailed next.

PDDL problem. For each new situation, the initial PDDL problem only contains the existing objects, locations and agents, and the initial state of the *standard* predicates. The *subgoal* predicates composing the goal and the *subgoal* action costs are determined by the LLM based on the current situation, as described in Sec. III-B and Sec. III-C, respectively. The outputs of the LLMs are translated into

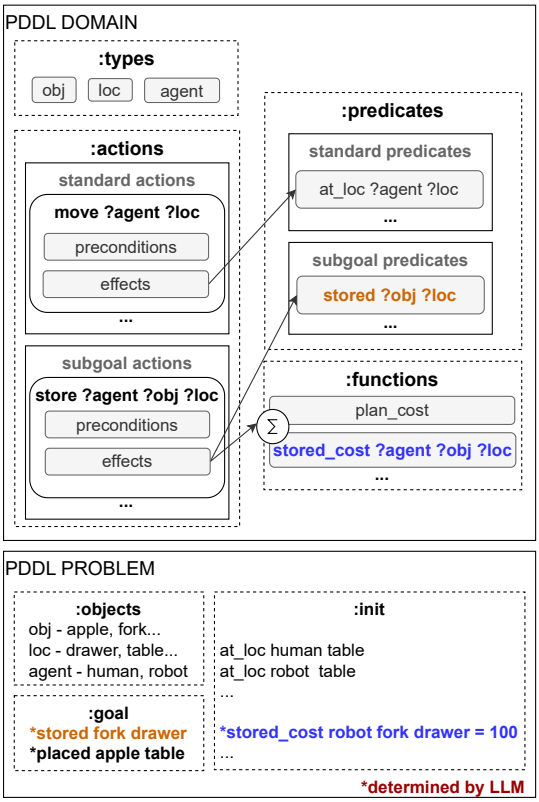

Fig. 4. PDDL planning definition: two types of actions have been defined in the planning domain (*subgoal* and *standard*). The *subgoal* actions directly achieve a subgoal, which is defined using a *subgoal* predicate. These actions have an associated cost. The *standard* actions are intermediate steps to reach the subgoals and have no cost. The planning problem contains the planning goal (made of subgoals) and the action costs, determined by the LLM.

PDDL elements in the following way:

PDDL goals. The subgoals returned by the LLM in the form of "object, predicate, location" are converted to PDDL goals, defined as *subgoal* predicates such as *"predicate ?obj ?loc"* (e.g., *"used_to_clean mop floor"*). The LLM is only allowed to reason about the *subgoal* predicates that constitute the planning goal, and cannot modify *standard* predicates.

PDDL action costs. The planner is responsible for distributing the actions in the plan to the appropriate agents. As it attempts to minimise the plan cost, it will avoid actions involving a high cost for an agent. The agent conditions should therefore be reflected in the action costs. Manually encoding the agent conditions in PDDL would be a complex and tedious task. We therefore take advantage of the LLM and map its recommendations to PDDL action costs. As an example, if the subgoal *"used_to_clean mop floor"* shall be disfavoured for the human, the corresponding cost *"used_to_clean_cost human mop floor"* is increased, and the planner will tend to avoid assigning the action *"clean mop floor"* to the human, as its effects increase the total cost by this value. Oppositely, for a subgoal that is favoured for one agent, the cost of its corresponding action will be increased for the other agents in the collaboration, thereby disfavouring the assignment of the favoured subgoal to these other agents. Decreasing the cost instead would lead to negative plan costs that the planner would be unable to deal with. With

this method, the automatic plan generation is now able to consider any agent condition in the action allocation.

IV. Evaluation

The system is evaluated in two stages. In Sec. IV-A, the results of generating planning subgoals from a list of NL abstract goals are presented. In Sec. IV-B, we evaluate the subgoal allocation for a list of NL agent conditions, on a number of scenarios with subgoals generated in the first evaluation. This second evaluation, therefore, assesses the full framework from the inputs (NL abstract goal and agent conditions) to the output (executable collaborative plan). The time taken for the process was evaluated on 320 runs, taking an average of 4.719 $\pm$ 4.208 seconds: where 3.143 seconds are for subgoal generation, 1.076 seconds for subgoal allocation recommendation, and 0.5 seconds for planning (based on a planning timeout, where a valid plan was returned in all cases). This result encourages the implementation of the system on real applications.

Scenario. The scenario evaluated consists in a kitchen setting, comprising 27 objects (cup, sponge, salmon...) and 17 locations (hob, fridge, table...). The available tasks are cooking, cleaning, storing, serving and placing.

Ground Truth. The datasets[1] created for the evaluation of the system contain a number of defined test cases, along with the expected ground truth (GT) output.

Metrics. The following metrics are used for evaluation:

1) *Completeness*: Number of generated subgoals corresponding to a GT subgoal, over the number of GT subgoals: $\frac{n\_correct\_subgoals}{n\_GT\_subgoals}$. If no subgoals should be generated and some are generated, completeness is 0.
2) *Correctness*: Number of generated subgoals corresponding to a GT subgoal, over the number of generated subgoals: $\frac{n\_correct\_subgoals}{n\_generated\_subgoals}$. For no subgoals generated, correctness is 1 if GT matches, 0 otherwise.
3) *Planning*: Whether a plan can be generated from the output subgoals. If the subgoals are not represented in the planning domain, a plan cannot be generated.
4) *Executability*: Whether the resulting plan is executable. A subgoal might be translated into a legal action such as cleaning the floor with an object. If this object is not valid for the goal (i.e., an apple), the plan is not considered executable.

A. Evaluation of Planning Subgoals Generation from an NL Abstract Goal

Dataset. A dataset containing 114 different NL goals has been created by adapting the data used in [3]. Their dataset varies in terms of time horizon and language complexity and was created via crowd-sourcing on Amazon Mechanical Turk, in-person kitchen user interviews, and benchmarks for everyday activities [33], [34]. Our work differs from theirs in two main aspects: we deal with a goal-focused collaboration rather than an instruction-driven assistive scenario, and we generate subgoals for posterior planning instead of using LLMs to generate a final plan. The instructions have been selected from the dataset and adapted into the form of a goal.

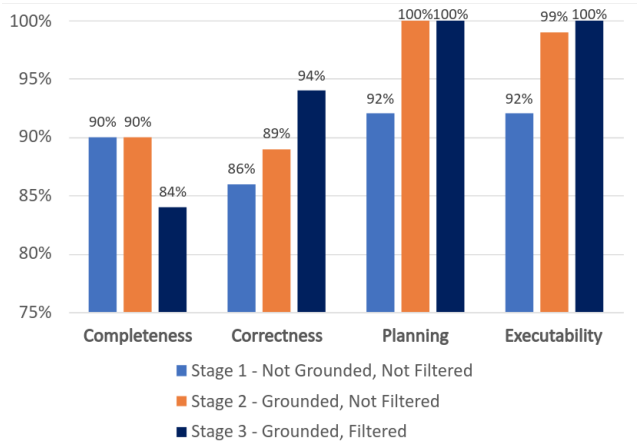

Fig. 5. Subgoal generation results: The correctness of the subgoals, plan success and executability are increased throughout the stages, whilst the completeness of the subgoals is decreased in stage 3. This represents a trade-off between completeness in the plan subgoals and final plan executability.

Furthermore, two new goal types have been added to deal with a wider range of abstract tasks: *multi-task* and *high-level abstraction*. Table I provides, for each goal type, an explanation, example, and expected output subgoals.

Results. Figure 5 shows the results of the subgoal generation at the three stages described in Sec. III-B. In stage 1, the LLM's raw response is considered without any grounding to the planning domain (happening in stage 2) or further filtering by the LLM (occurring in stage 3). As expected, subgoal correctness, planning success and executability are increased throughout the stages, whilst completeness of the subgoals decreases in stage 3. At stage 2, all the subgoals (with 89% correctness) are grounded to the domain and planning reaches 100%, whilst only 1% of the plans are not executable due to legal subgoals not making sense (e.g. bowl used to clean sink). At stage 3, after filtering the subgoals based on common sense and goal contribution through an LLM, correctness increases to 94% and all plans are executable, at the cost of a decrease in completeness (84%). The filtering stage greatly increases correctness, though a good compromise between completeness and executability needs to be found by tuning the filtering prompts. In most cases, our system proves to generate correct and executable HRC plans from an NL abstract goal. The subgoals are automatically defined by taking into account the current environment, eliminating the laborious step of manually defining a specific planning problem for each scenario.

B. Evaluation of Subgoal Allocation for Different NL Agent Conditions

Dataset. A dataset including 28 agent conditions has been created to evaluate the subgoal allocation adaptation to the collaborating agents. Conditions are divided into positive and negative, intended to result in some subgoals being favoured or disfavoured for the involved agent respectively. Table II provides, for each condition type, an example condition and its GT LLM output. Each condition is evaluated in four different scenarios (generated in the evaluation Sec. IV-A), resulting in a total of 112 tests evaluating the full system.

Goal Type (n examples)	Explanation	Example Goal	Ground Truth Output Subgoals
NL Single Primitive (4)	NL queries for a single primitive.	I want to let go of the coke can.	coke, placed, trash can
NL Nouns (25)	NL queries focused on abstract noun synonyms.	I want to serve something with caffeine at the table.	coffee, served, table
NL Verbs (25)	NL queries focused on abstract verb synonyms.	I want to prepare a chicken meal.	chicken, cooked, grill vegetables, cooked, hob
Structured Language (15)	Structured language queries.	I want to pick up the apple and move it to the trash.	apple, placed, trash can
Context/Disturbances (15)	Queries in unstructured formats.	My favorite drink is redbull, I want one. I am at the counter.	redbull, placed, counter
Long-Horizon (15)	Long-horizon queries that require many steps of reasoning.	I spilled my coke on the table, I want to throw it away and bring something to clean.	coke, placed, trash can cleaning cloth, used to clean, table
Multi-task (10)	Multi-task queries.	I want to clean the floor and cook a salmon.	mop, used to clean, floor salmon, cooked, hob
High-level abstraction (5)	No mention of specific objects or locations.	I want to tidy up the room.	spoon, stored, drawer cup, stored, cupboard banana, stored, fridge, etc.

TABLE I

GOAL TYPES USED FOR EVALUATION

Agent Condition Type	Example Condition	GT Output Subgoals to Favour/Disfavour
+ve Robot	The robot is fast at tidying up objects.	Favour robot: cup, stored, cupboard
+ve Human	The human loves to cook.	Favour human: salmon, cooked, hob
-ve Robot	The robot can't get wet.	Disfavour robot: mop, used to clean, floor
-ve Human	The human has back pain.	Disfavour human: cloth, used to clean, floor

TABLE II

EXAMPLE AGENT CONDITIONS USED TO EVALUATE THE SYSTEM

Agent Condition Type	Subgoals to favour		Subgoals to disfavour	
	Compl.	**Corr.**	**Compl.**	**Corr.**
+ve Robot	74%	75%	96%	96%
+ve Human	71%	71%	100%	98%
-ve Robot	96%	96%	72%	77%
-ve Human	96%	96%	75%	77%
All	**84%**	**84%**	**86%**	**88%**

TABLE III

RESULTS FOR SUBGOAL ALLOCATION REASONING

(COMPLETENESS AND CORRECTNESS)

Results. The results in Table III show how the system finds the right subgoals to favour in 84% of cases, and to disfavour in 86% of cases. Although there is no significant difference between the human and robot conditions, there exists a difference in performance for positive and negative conditions. Positive conditions involve some subgoals to favour and none to disfavour, whilst negative conditions involve some subgoals to disfavour and none to favour. The system demonstrates greater performance in determining that no subgoals should be favoured or disfavoured, achieving success in more than 96% of such cases, as opposed to the 71-77% success rate observed when it needs to identify particular subgoals to either favour or disfavour. In addition to further tuning of the LLM prompts, common-sense knowledge covering social interactions [35] could be injected into the LLM to increase the performance. Nevertheless, we demonstrate how LLMs have the potential to help robots reason about NL agent conditions during HRC plans, bringing in a large level of adaptability and naturalness in such scenarios.

V. LIMITATIONS AND FUTURE WORK DIRECTIONS

We identify a number of limitations of the system, presenting potential opportunities for future work. Firstly, the system inherits the limitations of LLMs, the performance of which significantly depends on the prompt given. Grounding and filtering have proven to get rid of incorrect LLM responses, but the trade-off between executability and flexibility in the system still remains. Retrieval Augmented Generation [36] could be implemented, integrating common-sense social, physical and eventive knowledge [35] into pretrained LLMs. A second limitation lies in the fact that the system reasons and plans based on subgoals that have no temporal dependencies between them. This, although limiting the complexity of the application, allows for a framework generalisable to other applications with any type of independent goals. Lastly, for an optimal workload allocation among the agents to be obtained, further research such as the one presented in [37] should be conducted to find "optimal" cost values for the favoured or disfavoured subgoals.

VI. CONCLUSIONS

In this paper, we have implemented a reasoning and planning framework, PlanCollabNL, that combines LLMs and AI task planning to improve HRC plan generation in two principal ways. Firstly, the planning problem definition is simplified and accelerated by leveraging LLM common-sense reasoning capabilities to automatically generate planning subgoals from an NL abstract goal. Secondly, a high degree of adaptability to the collaborating agents is achieved by influencing the allocation of the planning subgoals based on an LLM reasoning about NL agent conditions. The planning problem definition has been structured in a generalisable way, so that the system can be easily extended to a wide range of tasks. The framework has been evaluated for a number of goals and agent conditions, and is now ready for implementation in a real robot. Results show that executable and agent-adapted plans are successfully generated in most cases, eliminating the need for a manual and tedious definition of restricted and inflexible planning problems and agent models.

REFERENCES

[1] S. Izquierdo-Badiola, G. Canal, C. Rizzo, and G. Alenyà;, "Improved task planning through failure anticipation in human-robot collaboration," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7875–7880.

[2] S. Devin, A. Clodic, and R. Alami, "About decisions during human-robot shared plan achievement: Who should act and how?" in *Social Robotics*. Springer International Publishing, 2017, pp. 453–463.

[3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," in *Conference on Robot Learning*, Apr. 2022.

[4] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents," *ArXiv*, vol. abs/2201.07207, Jan. 2022.

[5] R. Lallement, L. De Silva, and R. Alami, "HATP: Hierarchical Agent-based Task Planner," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*, 2018.

[6] S. Bezrucav and B. Corves, "Improved AI planning for cooperating teams of humans and robots," in *Proceedings of the 8th ICAPS Workshop on Planning and Robotics (PlanRob)*, 2020.

[7] S. Devin and R. Alami, "An implemented theory of mind to improve human-robot shared plans execution," *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 319–326, 2016.

[8] G. Milliez, R. Lallement, M. Fiore, and R. Alami, "Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring," in *11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2016, pp. 43–50.

[9] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati, "Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change)," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, Nov. 2022.

[10] M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL—The Planning Domain Definition Language," 1998.

[11] M. Fiore, A. Clodic, and R. Alami, "On Planning and Task achievement Modalities for Human-Robot Collaboration," in *International Symposium on Experimental Robotics (ISER 2014)*, 2014, p. 15p.

[12] O. Görür, B. Rosman, G. Hoffman, and S. Albayrak, "Toward integrating theory of mind into adaptive decision- making of social robots to understand human intention," in *HRI Workshop on the Role of Intentions*, 2017.

[13] O. C. Görür, B. Rosman, F. Sivrikaya, and S. Albayrak, "Social cobots: Anticipatory decision-making for collaborative robots incorporating unexpected human behaviors," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2018, p. 398–406.

[14] O. C. Görür, B. Rosman, and S. Albayrak, "Anticipatory bayesian policy selection for online adaptation of collaborative robots to unknown human types," in *International Conference on Autonomous Agents and MultiAgent Systems*, 2019, p. 77–85.

[15] B. Zhang and H. Soh, "Large Language Models as Zero-Shot Human Models for Human-Robot Interaction," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7961–7968, Oct. 2023, conference Name: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) ISBN: 9781665491907 Place: Detroit, MI, USA Publisher: IEEE.

[16] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *36th Annual ACM Symposium on User Interface Software and Technology*, Oct. 2023, pp. 1–22.

[17] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "TidyBot: personalized robot assistance with large language models," *Autonomous Robots*, vol. 47, pp. 1–16, Nov. 2023.

[18] I. Dasgupta, A. K. Lampinen, S. C. Y. Chan, A. Creswell, D. Kumaran, J. L. McClelland, and F. Hill, "Language models show human-like content effects on reasoning," *ArXiv*, vol. arXiv:2207.07051, Jul. 2022.

[19] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," Microsoft, Tech. Rep. MSR-TR-2023-8, February 2023.

[20] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "PaLM-E: an embodied multimodal language model," in *40th International Conference on Machine Learning*, vol. 202, 2023, pp. 8469–8488.

[21] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "ProgPrompt: program generation for situated robot task planning using large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 999–1012, Aug. 2023.

[22] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner Monologue: Embodied Reasoning through Planning with Language Models," in *6th Conference on Robot Learning*, 2023, pp. 1769–1782.

[23] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex, "Planning With Large Language Models Via Corrective Re-Prompting," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, Nov. 2022.

[24] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models," in *IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2998–3009.

[25] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as Policies: Language Model Programs for Embodied Control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 9493–9500.

[26] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling, "PDDL Planning with Pretrained Large Language Models," in *NeurIPS 2022 Foundation Models for Decision Making*, Nov. 2022.

[27] Z. Zhao, W. S. Lee, and D. Hsu, "Large Language Models as Commonsense Knowledge for Large-Scale Task Planning," in *RSS 2023 Workshop on Learning for Task and Motion Planning*, Jun. 2023.

[28] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh, "Translating Natural Language to Planning Goals with Large-Language Models," *ArXiv*, vol. arXiv:2302.05128, Feb. 2023.

[29] Y. Ding, X. Zhang, S. Amiri, N. Cao, H. Yang, C. Esselink, and S. Zhang, "Robot Task Planning and Situation Handling in Open Worlds," *ArXiv*, vol. arXiv:2210.01287, Oct. 2022.

[30] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "LLM+P: Empowering Large Language Models with Optimal Planning Proficiency," *ArXiv*, vol. arXiv:2304.11477, May 2023.

[31] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *Computing Research Repository (CoRR)*, vol. abs/2005.14165, 2020.

[32] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[33] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 737–10 746.

[34] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. Vainio, Z. Lian, C. Gokmen, S. Buch, C. K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei, "BEHAVIOR:benchmark for everyday household activities in virtual, interactive, ecological environments," *Computing Research Repository*, vol. abs/2108.03332, 2021.

[35] J. D. Hwang, C. Bhagavatula, R. Le Bras, J. Da, K. Sakaguchi, A. Bosselut, and Y. Choi, "Comet-Atomic 2020: On Symbolic and Neural Commonsense Knowledge Graphs," in *AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 6384–6392.

[36] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *34th International Conference on Neural Information Processing Systems*, Dec. 2020, pp. 9459–9474.

[37] S. Izquierdo-Badiola, G. Alenyà, and C. Rizzo, "Adaptive human-robot collaboration: Evolutionary learning of action costs using an action outcome simulator," in *31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2023.