# PlanVerb: Domain-Independent Verbalization and Summary of Task Plans

**Gerard Canal,[1] Senka Krivić,[1] Paul Luff,[2] Andrew Coles[1]**

[1] Department of Informatics, King's College London
[2] King's Business School, King's College London
gerard.canal@kcl.ac.uk, senka.krivic@kcl.ac.uk, paul.luff@kcl.ac.uk, andrew.coles@kcl.ac.uk

## Abstract

For users to trust planning algorithms, they must be able to understand the planner's outputs and the reasons for each action selection. This output does not tend to be user-friendly, often consisting of sequences of parametrised actions or task networks. And these may not be practical for non-expert users who may find it easier to read natural language descriptions. In this paper, we propose PlanVerb, a domain and planner-independent method for the verbalization of task plans. It is based on semantic tagging of actions and predicates. Our method can generate natural language descriptions of plans including causal explanations. The verbalized plans can be summarized by compressing the actions that act on the same parameters. We further extend the concept of *verbalization space*, previously applied to robot navigation, and apply it to planning to generate different kinds of plan descriptions for different user requirements. Our method can deal with PDDL and RDDL domains, provided that they are tagged accordingly. Our user survey evaluation shows that users can read our automatically generated plan descriptions and that the explanations help them answer questions about the plan.

## Introduction

Plans produced by a task planner may not be easy to understand by non-expert users. This plan output, usually written as a sequence of parametrised actions, does not integrate enough information for users not familiar with the domain to understand it and the possible reasons for the plan's actions.

These users may be more familiar with natural language descriptions of the plans, narrated as a sequence of sentences describing the actions and involving the parameters. Furthermore, this narration of the plan can include causality information to link the actions together, making more explicit why each action was taken. We believe this would make it easier for those users to understand the plan, possibly increasing their trust in the planner. Additionally, this may also enable planning systems to narrate the plan themselves, fostering interaction with the user. A clear example of this would be that of a robot acting in human environments and explaining its plans to the users around.

In this paper, we present PlanVerb, a domain-independent method to verbalize task plans for planners based on PDDL (Fox and Long 2003) and RDDL (Sanner 2010). For this,we first propose semantic tagging for planning domains that specify the building blocks of the verbalized sentences (verb, subject, and complements). The tags are used by PlanVerb to generate the sentences, but may also be useful for readers of the domain to get a quick idea of what each action represents. We also present an action compression method to summarise plans by joining together actions that act on the same parameters. An example of this are compressions of navigation actions going through an intermediate point. Finally, we propose an extension to the verbalization space parameters from Rosenthal, Selvaraj, and Veloso (2016), previously used to narrate robot navigation. These parameters allow the generation of verbalizations at different levels of detail including only certain actions or objects, and with more or fewer causality explanations. A user evaluation with 42 participants has demonstrated that the proposed approach generates understandable plan verbalizations.



## Related Work

This work on task plan verbalization extends the work by Rosenthal, Selvaraj, and Veloso (2016), where verbalization is applied to the narration of mobile robot navigation routes. In that work, the authors introduce a verbalization space that covers the variability in utterances that can be used to describe the route to different users. The route and map of the robot are used to instantiate sentences that narrate the robot experience. They then performed a user study in Perera et al. (2016) where they analysed the kinds of questions that the users can request to the robots to obtain the desired explanations, to then learn a mapping between user queries and verbalization space parameters. This approach was further adapted in Zhu et al. (2017) to narrate manipulation tasks along with navigation, including PDDL actions. We have extended this notion of verbalization and verbalization spaces and applied it to task plans in a domain-independent fashion, integrating causality information to explain the relations between actions, with applications beyond robotics. Furthermore, we do not use pre-written sentence templates, but only tagging of the actions' syntactic elements.

Verbal communication of plans has been deemed necessary in robotic scenarios involving humans. Fiore et al. (2016) verbalize the actions in the plan for the user, explaining which actions will be executed and in what order. Canal et al. (2019) communicate the next action in the plan when the "inform" action is executed as part of the plan. Both works provide domain-dependent verbalization of the plans, probably written specifically for the task to be performed. In Singh et al. (2020), robot teams verbalize explanations of their actions and intentions to increase human understanding. The plan is verbalized by partitioning it based on the informativeness of the actions. The utterances come from predefined templates of possible phrases. Similarly, Nikolaidis et al. (2018) explore how utterances improve Human-Robot Collaboration with a robot that issues commands to users and explains why actions are done. The proposed formalism optimally combines verbal communications and robot actions to improve task performance. Neither of these works make causal relations between actions explicit, which may help users understand the reasoning behind the actions.

State verbalization was performed in Moon et al. (2019), where language descriptions of scene graphs are verbalized and used for scene understanding to describe the states while executing the plan, although these descriptions are not yet linked with the planning domain or planner.

Hayes and Shah (2017) explain robot control policies, verbalizing learned action conditions queried by the user. Similar to our domain description tagging, they add function decorators in the code to be able to verbalize the actions performed by the robot. Sridharan and Meadows (2019) present a theory of explanations for Human-Robot Collaboration. With it, they represent, reason, and learn knowledge to generate explanations, an explanation categorisation, and an explanation construction method. The defined characteristic axes can be seen as an equivalent of the verbalization space. Causal chains have been used to provide explanations in Seegebarth et al. (2012), where plans are represented in first-order logic with explanations being proofs based on causal links. Madumal et al. (2020) also use causal chains to generate explanations for RL agents using decision trees.

We summarise a plan by compressing some of the actions appearing in it. This is similar to work performed on Macro-Operators (Botea et al. 2005; Coles, Fox, and Smith 2007), where a set of actions is joined to form a macro-action. Similarly, we join sets of actions operating in intermediate parameters to verbalize them together. Other summarisation approaches, such as Myers (2006), summarise by describing features based on semantic concepts, while we compress redundant parts of the plan to show it as a whole.

## Semantic Domain Information Tagging

In order to generate sound sentences that represent each action and its parameters, we need information on how those actions relate to the parameters, and what do they mean.

For this, we propose to tag the domain file with information on how to generate sentences for each action. Thus, our method requires the input domains to be tagged with semantic information. While this introduces some manual work on the side of the domain expert, we believe it can also be useful to encourage commenting those domains, making it easier to understand the meaning of each action by the domain users. Therefore, we propose a commenting format to add semantic information to the actions. We denote these tags as "semantic information tags" as they will help the domain readers to understand the semantics of the action without the need of digging into its conditions and effects. The tags describe the syntactic information on the actions and their parameters.

We propose a flexible approach to obtain the necessary information to verbalize the actions in the domain. Instead of writing all the templated verbalization sentences, we tag each action and predicate with the *verb* that they represent along with its syntactic complements, and the *subject* of the action. These tags may include the parameters of the actions which will be replaced by their grounded value in the plan.

Our proposed format allows the specification of alternatives to produce richer verbalizations (i.e., synonyms), which are selected at random. Optional complements such as prepositional clauses may be flagged as required to prevent them from being omitted based on the verbalization space parameters (as detailed in the next section). Alternative forms of the syntactical clause are separated by a forward slash (/), while prepositional clauses can be flagged as required with an exclamation mark (!) at the end. Phrasal verbs can be added by putting the particle in parentheses such that only the non-parenthesised part will be conjugated. For instance, the phrasal verb "look for" would be defined as `; verb = look (for)`. Fig. 1 shows an example of tagged action and predicate with different verbal options.

These tags are then used to generate sentences for each action. Verbs are conjugated to the appropriate tense using `mlconjug3` (Diao 2022). Thus, our method can generate sentences in past, present, and future, allowing the planning system to update the plan verbalization while executing it.

## Task Plan Verbalizations

Following the definition from Rosenthal, Selvaraj, and Veloso (2016), we will define the *verbalization* of a task plan as the process that converts the plan into a natural language description. A natural language description of the plan may be easier to understand by a wider range of users, including non-experts in planning nor the domain. This understanding can then be key to improve plan transparency and user trust, as users' acceptance can increase when the reasons for the system's actions are explained (Koo et al. 2015).

We propose a verbalization method that is domain-independent provided that the input domain has been tagged as described above. We use the ROSPlan system (Cashmore et al. 2015) as planning framework. This allows us to have a planner-agnostic method, as well as to support both PDDL-based and RDDL-based planners (by using the probabilistic extension by Canal et al. (2019)). In the case of RDDL, one caveat is that we are constrained to the subset of it supported by ROSPlan. Thus, causality information and goals (if present) may not be properly captured by ROSPlan, restricting the verbalization that our method can perform. We support durative and non-durative actions (PDDL2.1) but not processes or events (PDDL+).

### Verbalization Space

Different users will have distinct preferences or needs when it comes to obtaining task plan descriptions. An expert user

```
; verb = go / travel / move
; subject = ?v
; prep = from the ?from
; prep = to the ?to / towards the ?to !
(:durative-action goto_waypoint
 :parameters (?v - robot ?from ?to - waypoint)
```

(a) Example of PDDL action tagging.

```
; The robot ?r is at the waypoint ?wp
; verb = be
; subject = ?r
; prep = at the ?wp
(robot_at ?r - robot ?wp - waypoint)
```

(b) Example of PDDL predicate tagging.

Figure 1: Examples of semantic tags. A RDDL example can be found in the supplementary material[1].

may need a detailed, step by step description of the plan to find incongruities or erroneous actions. A lay user, instead, may prefer to read a summarised version of the plan, know what was performed to achieve the main goals, or get a summary of the actions that were applied on a particular object.

To cope with these different verbalization use cases, we have extended the concept of *verbalization space* suggested by Rosenthal, Selvaraj, and Veloso (2016) to cover the narration task plans. The verbalization space specifies different variations of the descriptions of the plans to cover different user preferences. It includes four parameters: abstraction, locality, specificity, and explanation, as detailed below.

The combination of the different parameters allows to generate various plan descriptions, from more detailed to more abstract and summarised, covering a wide range of situations. This verbalization space for task plans should be general enough for most use-cases, but can easily be extended to handle more parameters or combinations of them.

**Abstraction** The abstraction parameter $a \in A$ represents the level of concretion used in the verbalization of the plans. We consider four levels of abstraction:

*A1* No abstraction. This means that the verbalization will include numerical values such as real-world coordinates of objects or locations. It also includes the duration of the actions (if available), as well as all their parameters. For this level, an extra file with the mapping from object instances to real-world data can be provided.

*A2* In this level, the parameter names are used instead of the available real-world values. It still verbalizes action durations and all the parameters, as well as intermediate values for compressed actions.

*A3* The duration of the actions is not verbalized, while all the parameters and intermediate values (such as via points) for compressed actions are kept.

*A4* In the most abstract level, only the essential parameters of the actions are verbalized, which are those needed for a grammatically correct sentence and those flagged as required. Intermediate values are also skipped.

**Locality** The locality parameter $l \in L$ narrows the verbalization scope, to base it only on points of interest of the user or a range of actions. We define three values for the locality:

*All plan* All the actions in the plan are verbalized.

*Range of actions* Restricts the scope to a subset of the actions of the plan. For instance, the verbalization would only take from the third action to the fifteenth one.

*Action or object* Limits the verbalization to those actions including a specific object instance as a parameter, or verbalizes all the actions with a given name.

**Specificity** The specificity parameter $s \in S$ describes how specific the description of the plan should be regarding the level of detail. It includes three options:

*General picture* A generic description of the main highlights of the plan. It focuses on actions achieving goals, and verbalizes these along with their justifications, provided that they are set so by the explanation parameter.

*Summary* The verbalization will compress actions when possible, giving a more compact representation. These compressions short-cut actions that act on intermediate objects (e.g. navigation via intermediate points), or join actions that are repeated with different objects/subjects. This is further detailed in the next section.

*Detailed narrative* Generates a detailed description of the plan without summarising nor compressing any action. Thus, all the actions will appear in the plan narration.

**Explanation** The explanation parameter $e \in E$ specifies the level of justifications between actions that will be narrated. We have considered three kinds of verbalizable justifications: immediate justifications of actions, deferred justifications of actions, and goal-achieving explanations.

An action $a_j$ is an *immediate justification* of another action $a_i$ if $\forall k \in [i..j)$, there is a causal link between $a_k$ and $a_j$, where $i$, $j$, and $k$ are the indices in which the actions appear in the original plan. Thus, $a_j$ will be an immediate justification of all the actions in $[a_i..a_j)$, which are the actions that allow $a_j$ to happen. A *deferred justification*, instead, happens when an action $a_i$ has a causal link with a non-consecutive action $a_j$. Therefore, we have a deferred justification when $\exists k \in [i..j)$ such that $a_k$ does not have a causal link with $a_j$. *Goal-achieving explanations* make goal achievement explicit, showing when an action was taken to complete a specific goal. The explanation levels are:

*E1* No explanation is verbalized, so actions are verbalized sequentially in order of appearance in the plan.

*E2* Joins actions when one action is an immediate justification of another action, and verbalizes them making the causality between the actions explicit.

*E3* Adds deferred justifications for actions that have a causal link with another action that appears later in the plan, but only if the action that is being justified achieves a goal. Deferred justifications to actions that act as an immediate justification are not verbalized.

[1]Available at https://bit.ly/planverb-supplementary

```
(grasp r A)
(grasp r B) → (grasp r [A B]) → (grasp r [A B C])
(grasp r C)     (grasp r C)
```

(a) Compression with multiple objects. The resulting action symbolizes “r will grasp A, B, and C”.

```
(goto r A B)     (goto r A C)        (goto r A D)
(goto r B C) →   (through B)    →    (through B and C)
(goto r C D)     (goto r C D)
```

(b) Compression with intermediate parameters. The resulting action means “r will go from A to D”.

Figure 2: Action compressions for plan summarisation examples. More can be found in the supplementary material[1].

*E4* The explanations of the goals that are achieved by the actions are added to the verbalization, along with the explanations from the lower levels.

*E5* Includes all deferred justifications (for all the causal links of an action).

## Plan Summarisation through Action Compression

It is often the case with some domains that the same action is sequentially repeated throughout the plan, with the in-between appearances of the action providing intermediate values that may not be very informative to the user.

Examples of this include navigation actions for a robot, where it can only move between a waypoint and another one connected to it. Thus, to reach a certain position, it must traverse a set of these waypoints, generating many actions that reach intermediate positions. Similarly, a consecutive sequence of the same action applied to different objects can be summarised as the action applied to the set of objects.

We propose an action compression method to deal with these kinds of actions to generate shorter plan verbalizations. We only compress actions in the aforementioned cases, and when there is only one free parameter (i.e., a grounded parameter whose value does not appear in both actions). Still, the method is easily extendable to more complex situations.

Given two consecutive appearances of an action in the plan, we compare their grounded parameters to create a pattern that indicates if each parameter had the same value in both actions, or the same instance appeared in different parameter positions. We then perform compression as follows:

- When all the action parameters but one have the same values in the same position, the resulting compression keeps those parameters and joins the free parameter in a list. Note that this method will compress parameters acting as an object or a subject. See example in Fig. 2a.
- When the same grounded parameter appears in different positions in both actions, we consider it as an intermediate parameter. The resulting compression removes the intermediate parameter and joins both actions by keeping the rest of parameters. To do so, the space left by the intermediate parameter is filled by the grounded values appearing at the same place in the other action. The intermediate parameters are kept to be used with abstraction levels 1-3. Fig. 2b shows an example of this compression.

The compression method starts at the beginning of the plan and checks every pair of consecutive actions trying to compress them according to the above procedure. When two actions are compressed, the resulting action is compared with the next one, extending the compression to the subsequent actions in the plan. The compressed action duration is computed as the time overlap between the two actions.

## The PlanVerb Algorithm

The plan is preprocessed and stored in an intermediate structure to later allow the generation of the verbalized sentences. This structure is a script of the plan to be verbalized. Each element $s_l \in V$ in the script $V$ is a 4-tuple $s_l = \langle a_i, I_{a_i}, D_{a_i}, G_{a_i} \rangle$, where $a_i$ is an action, $I_{a_i}$ is a list of immediate justifications (actions $a_j$ with a causal link *to* $a_i$), $D_{a_i}$ is a list of deferred justifications (actions $a_k$ with a causal link *from* $a_i$), and $G_{a_i}$ a list of goals achieved by $a_i$.

We first compute the action causality chains from the plan. For this, we use a graph-based representation of the plan, such as the one from Lima et al. (2020) that is integrated into ROSPlan. From the plan graph, we compute causality chains for those actions achieving a goal by traversing the graph's causal edges from these goal-achieving actions backwards.

Algorithm 1 shows the pseudocode of the PlanVerb algorithm. To start, the actions in the plan are compressed using the ComputePlanCompressions method (line 5), as described in the section above. The compression method splits the plan into one plan per each subject performing an action. This enhances the number of action compressions, as only actions appearing consecutively in the plan are compressed.

Then, the causality chains are used to generate a full plan script integrating every action's information (immediate justifications, deferred, and goals). We call this script the causality script, and it is generated in ComputeCausalityScript (line 2). Justifications are also considered on a per-subject basis, as immediate justifications may not be consecutive in the full plan, but be in the subject-split plan.

The causality script is then iterated and the verbalization space parameters are applied to generate a verbalization script including the actions that will be finally verbalized. Actions and action justifications are filtered based on the verbalization space parameters (lines 11–14). Actions acting as immediate justification are skipped and not included in the script, given that they will be verbalized with the action they support. Actions appearing in a deferred justification are not skipped. Instead, they are verbalized both as a (later) consequence of the causing action, and as an action with its own justifications when it appears later in the plan.

To avoid overcluttering the sentences, deferred justifications are skipped when they justify a skipped action (i.e., it acts as immediate justification to another action), or when they appear in a sentence where a goal is verbalized and the explanation level is lower than 5 (so, goals take precedence).

**Sentence generation** Each action in the script is verbalized in line 14 of Algorithm 1. The GenerateSentence method checks whether there are immediate, deferred justifications, or goals in the script, verbalizes each of them and joins them with pre-defined sentence linkers. The selected

**Algorithm 1:** The PlanVerb algorithm

**Input:** Plan $\pi$; Causality chains $C$; Semantic tags $T$
Verbalization space $(a, l, s, e) \in (A, L, S, E)$
**Output:** Verbalization $v$

```
GA := GetGoalAchievingActions(C)
CS := ComputeCausalityScript(π, C)
PC := []; v := []
if s == Summary then
    PC := ComputePlanCompressions(π, GA)
else if s == General Picture then
    CS := GetGoalAchievingScripts(GA, CS)
foreach c ∈ CS do
    if NotInLocality(c, l) then skip c
    else // Filter the scripts according to e
        c.I := FilterImmediateJustifications(c.I, e)
        c.D := FilterDeferredJustifications(c.D, e)
        c.G := FilterGoals(c.G, e)
        v.add(GenerateSentence(c, a, T, π, PC))
return v
```

linker is chosen at random, and the actions are verbalized to the appropriate tense based on the structure of the linker and the tense of the main action in the script.

The script may be tensed in future, past, or present depending on the execution point of the plan. The justifications and deferred justifications are tensed accordingly, with the verbs conjugated using the `mlconjug3` library. All the actions and predicates (goals) are verbalized similarly, taking the form of `subject + verb + indirect-object + direct-object + prepositional clauses`. Only the available parts of the sentence are used, based on the semantic tags of the action and the abstraction parameter.

The sentence generation process also checks whether the actions in the script are compressed and uses the compressed version, adding the intermediate values as via points and the action duration when specified by the level of abstraction.

### Verbalization Questioning

While the proposed verbalization approach is flexible, covering many kinds of user preferences, there may be cases where more information on a specific action is required. This could be done by setting verbalization space parameters accordingly (i.e. with a narrow locality), but here we propose a more flexible approach to question the verbalization to get information on a single action, also using natural language.

We have used spaCy (Honnibal et al. 2020) to parse partially grounded questions, which are then matched with the PDDL plan. Ambiguities are solved by asking back the user providing options on the remaining parameters to be grounded. Once an action has been matched, it is then verbalized using PlanVerb. In this case, all the deferred justifications and goals are verbalized.

## Evaluation

We have evaluated the proposed plan verbalization method and spaces. First, we provide some examples of automatically verbalized actions. Then, we analyse the impact of the verbalization space parameters. Finally, we comment on the results of an online survey regarding the verbalization.

For our evaluation we used ROSPlan with the POPF planner (Coles et al. 2010) for PDDL domains, and the PROST planner (Keller and Eyerich 2012) for RDDL domains[2] .

### Examples of Verbalized Actions

Here we will present some verbalized actions produced by our algorithm. A wider set of examples can be found in the supplementary material[1]. In this section, we use a robotics domain where mobile robots perform navigation, pick, place, and handover tasks. The exemplified plans include two robots: the narrator (in first person) and "Tomo".

In the following examples, black sentences refer to the main action, blue sentences to immediate justifications, green sentences to deferred justifications, and red to goals. Sentences appear in different tenses to show that the method can generate sentences at different points of execution.

**Example 1: Abstraction** We start with an action appearing early in the plan where Tomo locates the manager. This action enables the actions of "request person" and "give object", being the latter achieved by the other robot at the last part of the plan. The sentence verbalized with $(a, l, s, e)$ = (A3, All plan, Summary, E4), action durations are not included, and all the parameters are verbalized.

> Tomo will locate the manager, which will allow me to **later** request the manager at the kitchen corridor and me to hand post2 to the manager at the kitchen corridor.

If verbalized with abstraction A4, the resulting sentence ignores the location prepositional clause:

> Tomo is going to locate the manager, which will allow me to **later** ask the manager and me to deliver post2 to the manager.

**Example 2: Specificity** This shows action compression. With verbalization space parameters $(a, l, s, e)$ = (A3, All plan, Summary, E4), intermediate actions are compressed:

> Tomo will travel from the kitchen shelf towards the kitchen counter (via coffee table) so Tomo can leave the paper at the kitchen counter to achieve the goal of the paper being at the kitchen counter.

When using Detailed Narrative instead, the sentence is as follows (now in past tense). Note that as there is no compression, actions from different subjects are interleaved:

> Tomo is going to go from the kitchen shelf to the coffee table. I am going to go from the kitchen table towards the kitchen corridor, which will allow me to go from the kitchen corridor towards the microwave. Tomo will travel from the coffee table to the kitchen counter so Tomo can put the paper at the kitchen counter to fulfill the goal of the paper being at the kitchen counter.

In the following example from the IPC'02 Rovers domain (Long and Fox 2003), subjects are compressed. With verbalization space parameters $(a, l, s, e)$ = (A2, All plan, Detailed Narrative, E1), the verbalization is:

[2] The code, domains, and the complete set of verbalized plans with all the combinations of verbalization space parameters can be found in https://github.com/gerardcanal/task_plan_verbalization

Rover3 will travel from waypoint7 towards waypoint0 (taking 5 seconds). Rover2 is going to travel from waypoint7 towards waypoint0 (taking 5 seconds).

When compressed with the Summary specificity parameter and set to present tense, it becomes:

Rover3 and Rover2 are traveling from waypoint7 towards waypoint0 (taking 5 seconds).

**Example 3: Immediate and deferred explanations** Finally, an example of verbalization with both immediate and deferred justifications. Here, the verbalization parameters are $(a, l, s, e)$ = (A4, All plan, Summary, E4):

Tomo is going to move to the office entrance 1, which will allow Tomo to grasp the post1 so Tomo can **later** leave the post1 at the main office desk.

**Example 4: Verbalization questioning**

Q: Why did you grasp the paper?
**A: I took the paper at the desk, which will allow me to later place the paper at the office shelf.**

Q: Why did you go to kitchen?
**A: Do you refer to when I will go from the entrance or the corridor towards the kitchen?**
Q: From the entrance.
**A: I am going from the entrance towards the kitchen to later be able to check the kitchen, find person1, locate person2, and move from the kitchen to the desk.**

## Effect of the Verbalization Space Parameters

We have validated the effect of the verbalization space parameters with a set of test domains. Those include the office robot domain (4 instances), the IPC'02 Rovers domain (Long and Fox 2003) (19 instances), the IPC'08 CrewPlanning domain (Barreiro, Jones, and Schaffer 2009) (30 instances) for PDDL. For RDDL, we have used the IPPC'14 triangle tireworld (Little and Thiebaux 2007), the print-fetching domain from (Canal et al. 2019), and 3 interactive robotics domains (Canal, Torras, and Alenyà 2022), involving assistive feeding and dressing tasks. We have computed a plan for all the domains and instances and verbalized it with all the combinations of parameters. Fig. 3 shows the average number of words for all the verbalized plans and parameters.

Regarding the abstraction parameter (Figs. 3a and 3b), the figures show that the higher the level of abstraction, the fewer the number of verbalized words. Note that for abstraction A1, in this experiment we have sampled random real coordinates from a 2D space to represent the locations appearing in the problem instances.

For the explanation parameter (Figs. 3b and 3c), the number of words increases with the level of explanation, as the text becomes more verbose. E2 has a slight increase, as the same number of actions are verbalized but linked together. The deferred justifications added by level E3 increase more the number of words, surpassed by E4 with the verbalized goals. Lastly, the inclusion of all the deferred justifications in level E5 generates the longest verbalizations.

Specificity (Figs. 3a and 3c) also has a clear effect on the word number. The General Picture is the most summarised one, including only some actions. The Summary level includes all the actions but compresses some of them, producing shorter narrations than the Detailed Narrative.

## Online User Survey

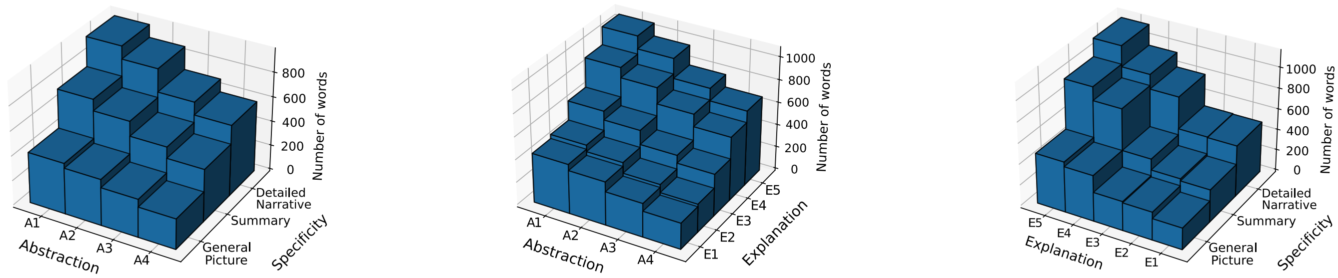
We have conducted an online survey to assess both the usefulness of the provided explanations and the understandability of the generated sentences. The survey was answered by 42 people in two groups. Two verbalizations were shown to each user. The first one, $v_1$ is a step-by-step plan of two robots, Tomo and Asro, performing tasks of the office domain. The narration for $v_1$ was generated with parameters $(a, l, s, e)$ = (A3, All plan, Detailed Narrative, E1). The other one, $v_2$ is a summarised version of the same plan including explanations, generated with parameters $(a, l, s, e)$ = (A3, All plan, Summary, E4). One group would see first the step-by-step plan $v_1$ and then the summarised one $v_2$; the other would see them in the opposite order. The background of the users ranged from robotics, computer science, AI planning, and unrelated disciplines (non-technical). 57% of the users were not familiar with task planning (lay users), and the 28% had occasionally seen or used a task planner before (non-experts). Four users were considered expert. We kept them in the analysis because, while our focus is on non-expert users, we wanted to see if there were notable differences in views or comments from them. We did not find differences in performance, while they provided meaningful opinions. All of the users were fluent in English. The survey involved multiple-choice questions on their opinions on why they thought some specific actions were appearing in the plan based on the goals of the robots. The multiple-choice questions were followed by 5-point and 7-point Likert-scale questions regarding their agreement with different statements on plan understandability. Finally, some open-ended questions concluded the survey, available in the supplementary material[1]. We have used a confidence level of 95% in the statistical tests used to analyse the survey results.

An F-test showed there were no significant differences between the two groups, for which the following results will aggregate the answers regarding $v_1$ and $v_2$ for both groups.

The answers to the multiple-choice questions were given one point for a correct value, and half a point for partially correct answers (for instance, when the answer involved two reasons but only one of them was selected). Our results clearly show that $v_2$, which included justifications, helped the users to better answer the questions. More than 80% of the users were able to answer correctly, with the highest question being 97.62%. In contrast, for the step-by-step description $v_1$ (without justifications) only half of the users gave a correct answer, with the maximum for a single question being 61.90%. For each question, between the 20% and 40% of the users stated they did not know the answer for $v_1$, while this percentage was at most 2.38% for $v_2$. We have assessed the significance of these results with a $\chi^2$ test.

Regarding the Likert questions, users were more confident in their responses for $v_2$ ($\tilde{x} = 5.67$ out of 7)[3] than for $v_1$ ($\tilde{x} = 4.11$ out of 7). On how easy it was to answer, the

[3]Where $\tilde{x}$ represents the arithmetic mean.




(a) Average number of words for abstraction and specificity.

(b) Average number of words for abstraction and explanation.

(c) Average number of words for explanation and specificity.

Figure 3: Effect of the verbalization space parameters in the average number of generated words in the verbalization.

means were $\tilde{x} = 4.02$ out of 5 for $v_2$ and $\tilde{x} = 2.45$ for $v_1$. Those results indicate that users found explanations to be helpful to answer the questions, clearly shown by the answers to how quickly they could find the reasons behind each questioned action: $\tilde{x} = 4.12$ out of 5 for $v_2$ and $\tilde{x} = 2.30$ for $v_1$. Users also found the descriptions in $v_2$ easier to read and understand. Similarly, they reported higher satisfaction with the plan description for $v_2$, which included explanations. A t-test found statistical significance for all these answers.

When asked about the grammatical soundness of the generated sentences, there were no significant differences between both verbalizations. For $v_1$, $\tilde{x} = 4.19$ out of 5, while $v_2$ got a $\tilde{x} = 4.17$. Thus, both representations generated by PlanVerb were found to be correct and readable.

When asked if the description of the plan made it easier to answer the questions, most users agreed for $v_2$ (with justifications) against $v_1$ (without justifications). Users that saw $v_2$ last agreed it helped more than $v_1$, while those seeing $v_1$ last disagreed that $v_1$ helped more than $v_2$. This supports the claim that justifications help users understand the reasons for actions. A t-test also showed significance for these results.

When asked about improvements, some users pointed to elements that can already be solved by the different combination of verbalization space parameters, such as showing only actions achieving goals. Some users believed there was too much information, with excessive granularity in $v_1$, while others mentioned $v_1$ was missing information while $v_2$ wasn't. This appears in different answers and suggestions such as adding temporal information (which we can do with different parameters). Therefore, users' answers clearly support the need for different parametrizations, given that users will have their preferences over the best verbalization. A few users mentioned that one plan for each robot would be easier to understand, which supports the idea of joining plans by subject and summarizing them separately as we propose. Finally, many users suggested adding visualisation along with the verbalization, with ideas we leave for future work.

The conducted survey demonstrates that PlanVerb generates grammatically sound narrations of task plans that make sense to users. The users' answers further support the need for different kinds of verbalizations, which can be achieved with the verbalization space parameters we have proposed.

## Conclusions

In this paper, we have presented PlanVerb as a domain-independent method to automatically verbalize task plans. We have proposed a semantic tagging for PDDL/RDDL actions and predicates that provide the necessary information for PlanVerb to generate natural language sentences. Then, by using causality information between actions, we are able to generate sentences that make this causality explicit, both for immediate justifications of actions appearing consecutively in the plan and for deferred justifications of actions that appear at a later stage. The narrated plans can also be summarised by compressing related actions. We have further extended the concept of verbalization space introduced by (Rosenthal, Selvaraj, and Veloso 2016) to cover task plans, adding a new explanation parameter that manages the amount of verbalized justifications. We have also introduced filtering by object or action in the locality parameter.

We have shown examples of verbalized sentences and evaluated the effect of the verbalization space parameters in different domains. The supplementary material[1] includes an extended set of examples. Finally, we have conducted an online survey where users were shown examples of verbalized plans. All users were able to read them and confirmed the sentences were grammatically sound. Moreover, the justifications helped them understand the plan, which supports the hypothesis that verbalizing causal chains fosters plan understanding. We believe this is a good step towards making task plans more understandable. However, users also pointed to the need for better Explainable Planning (XAIP) methods able to explain the underlying reasons for the actions beyond making causality explicit.

Although we can successfully verbalize plans that are understandable by users, some improvements may be done as future work. First, using some natural language processing techniques to improve sentence generation. Pronominalisation could help to make sentences more natural avoiding subject repetition, as well as pluralization of nouns (i.e., after some action compressions). Finally, the addition of preconditions and effects could be beneficial to the verbalization process, along with improved justification selection. This could be accompanied by plan visualization techniques to clarify the steps involved in the plan.

## Acknowledgments

This work has been supported by the EPSRC grant THuMP (EP/R033722/1), and 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. The authors would like to thank Mr Sekou Diao for his insights and help with the use of `mlconjug3`, and Mr Ionut Moraru, Mr Alexander Ortiz de Guinea, Dr Michael Cashmore, Dr Rita Borgo, and Dr Xavier Ferrer for fruitful discussions.

## References

Barreiro, J.; Jones, G.; and Schaffer, S. 2009. Peer-to-peer planning for space mission control. In *2009 IEEE Aerospace conference*, 1–9. IEEE.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*, 24: 581–621.

Canal, G.; Alenyà, G.; and Torras, C. 2019. Adapting robot task planning to user preferences: an assistive shoe dressing example. *Autonomous Robots*, 43(6): 1343–1356.

Canal, G.; Cashmore, M.; Krivić, S.; Alenyà, G.; Magazzeni, D.; and Torras, C. 2019. Probabilistic Planning for Robotics with ROSPlan. In *Towards Autonomous Robotic Systems*, 236–250. Springer International Publishing.

Canal, G.; Torras, C.; and Alenyà, G. 2022. Generating Predicate Suggestions based on the Space of Plans. An Example of Planning with Preferences. *User Modeling and User-Adapted Interaction*.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera Viñas, A.; Palomeras Rovira, N.; Hurtós Vilarnau, N.; and Carreras Pérez, M. 2015. Rosplan: Planning in the robot operating system. In *International Conference on Automated Planning and Scheduling*, 333–341.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *International Conference on Automated Planning and Scheduling*, 42–49.

Coles, A.; Fox, M.; and Smith, A. 2007. Online Identification of Useful Macro-Actions for Planning. In *International Conference on Automated Planning and Scheduling*, 97–104.

Diao, S. 2022. `mlconjug3`. https://github.com/SekouDiaoNlp/mlconjug3. Accessed: 2022-03-01.

Fiore, M.; Clodic, A.; and Alami, R. 2016. On Planning and Task Achievement Modalities for Human-Robot Collaboration. In *14th International Symposium on Experimental Robotics*, 293–306. Cham: Springer International Publishing. ISBN 978-3-319-23778-7.

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

Hayes, B.; and Shah, J. A. 2017. Improving robot controller transparency through autonomous policy explanation. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 303–312. IEEE.

Honnibal, M.; Montani, I.; Van Landeghem, S.; and Boyd, A. 2020. spaCy: Industrial-strength Natural Language Processing in Python. https://spacy.io/. Accessed: 2022-03-01.

Keller, T.; and Eyerich, P. 2012. PROST: probabilistic planning based on UCT. In *International Conference on Automated Planning and Scheduling*, 119–127.

Koo, J.; Kwac, J.; Ju, W.; Steinert, M.; Leifer, L.; and Nass, C. 2015. Why did my car just do that? Explaining semi-autonomous driving actions to improve driver understanding, trust, and performance. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 9(4): 269–275.

Lima, O.; Cashmore, M.; Magazzeni, D.; Micheli, A.; and Ventura, R. 2020. Robust Plan Execution with Unexpected Observations. In *ICAPS Workshop on Integrated Execution (IntEx) / Goal Reasoning (GR)*.

Little, I.; and Thiebaux, S. 2007. Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*.

Long, D.; and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20: 1–59.

Madumal, P.; Miller, T.; Sonenberg, L.; and Vetere, F. 2020. Distal Explanations for Explainable Reinforcement Learning Agents. *arXiv preprint arXiv:2001.10284*.

Moon, J.; Magazzeni, D.; Cashmore, M.; Buksz, D.; Lee, B.-H.; Moon, Y.-S.; and Roh, S.-H. 2019. Towards Explanations of Plan Execution for Human-Robot Teaming. In *Intl. Workshop on the Semantic Descriptor, Semantic Modelingand Mapping for Humanlike Perceptionand Navigation of Mobile Robots toward Large Scale Long-Term Autonomy*.

Myers, K. L. 2006. Metatheoretic Plan Summarization and Comparison. In *International Conference on Automated Planning and Scheduling*, 182–192.

Nikolaidis, S.; Kwon, M.; Forlizzi, J.; and Srinivasa, S. 2018. Planning with verbal communication for human-robot collaboration. *ACM Transactions on Human-Robot Interaction (THRI)*, 7(3): 1–21.

Perera, V.; Selveraj, S. P.; Rosenthal, S.; and Veloso, M. 2016. Dynamic generation and refinement of robot verbalization. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 212–218.

Rosenthal, S.; Selvaraj, S. P.; and Veloso, M. 2016. Verbalization: Narration of Autonomous Robot Experience. In *International Joint Conference on Artificial Intelligence*, volume 16, 862–868.

Sanner, S. 2010. Relational Dynamic Influence Diagram Language (RDDL): Language Description. http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf. Accessed: 2022-03-01.

Seegebarth, B.; Müller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *International Conference on Automated Planning and Scheduling*.

Singh, A. K.; Baranwal, N.; Richter, K.-F.; Hellström, T.; and Bensch, S. 2020. Verbal explanations by collaborating robot teams. *Paladyn Journal of Behavioral Robotics*, 12(1): 47–57.

Sridharan, M.; and Meadows, B. 2019. Towards a Theory of Explanations for Human–Robot Collaboration. *KI-Künstliche Intelligenz*, 33(4): 331–342.

Zhu, Q.; Perera, V.; Wächter, M.; Asfour, T.; and Veloso, M. 2017. Autonomous narration of humanoid robot kitchen task experience. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, 390–397. IEEE.