


## Article

# Coordination of Multiple Autonomous Agents Using Naturally Generated Languages in Task Planning

Li Wang \*  and Qiao Guo \*

School of Automation, Beijing Institute of Technology, Beijing 100081, China

\* Correspondence: wangli2015@bit.edu.cn (L.W.); qguo@bit.edu.cn (Q.G.)

Received: 24 July 2019; Accepted: August 23 2019; Published: 1 September 2019



**Abstract:** Language plays a prominent role in the activities of human beings and other intelligent creatures. One of the most important functions of languages is communication. Inspired by this, we attempt to develop a novel language for cooperation between artificial agents. The language generation problem has been studied earlier in the context of evolutionary games in computational linguistics. In this paper, we take a different approach by formulating it in the computational model of rationality in a multi-agent planning setting. This paper includes three main parts: First, we present a language generation problem that is connected to state abstraction and introduce a few of the languages' properties. Second, we give the sufficient and necessary conditions of a valid abstraction with proofs and develop an efficient algorithm to construct the languages where several words are generated naturally. The sentences composed of words can be used by agents to regulate their behaviors during task planning. Finally, we conduct several experiments to evaluate the benefits of the languages in a variety of scenarios of a path-planning domain. The empirical results demonstrate that our languages lead to reduction in communication cost and behavior restriction.

**Keywords:** multi-agent systems; task planning; communication; language generation; autonomous agents; coordination; state abstraction

## 1. Introduction

Compared with single-agent systems, multi-agent systems have the distribution properties of time, space, and function, and have several advantages in task applicability, execution efficiency, and system robustness [1]. Real-world applications of multi-agent systems include logistics [2], construction [3], search and rescue [4], warehouse automation [5], infrastructure placement [6], computer animation [7], etc. Due to the lack of complete knowledge, agents usually need to exchange their states, actions, or goals to collectively carry out system tasks. Consequently, a communication language or protocol should be predefined to inform cooperative strategies when designing multi-agent systems.

Two methods are commonly used to construct languages for agent communication. One is to design a certain artificial language for agents [8,9]. The other is to let agents communicate in natural languages [10,11]. Most of the studies on multi-agent planning and distributed control use the former method to exchange messages. The latter approach is helpful for human partners to understand the behavior of agents. However, agents must learn two different internal representations of themselves and humans, which can be counterproductive. In fact, it is not necessary for agents to use human languages in a situation where only artificial agents exist. In this paper, we try to create the agents' own languages that can be used to coordinate them. The languages are not predefined case-by-case and are naturally generated based on the abstraction of agents' states in the environment.

Our work is motivated by a fundamental question from agent coordination: what kind of information do agents need to communicate? We set out to answer the question by considering a particular situation in which communication is only available in task planning phase. In this case, the

common way for agents is to send a feasible plan to their teammates to be followed when cooperation is needed. The objective of this work is to construct a kind of language that can be used to specify plans, while not bringing many constraints for agents and reducing the cost of communication as much as possible.

The contributions of this work have three aspects: First, we formulate a language generation problem for multi-agent systems and introduce some fundamental features of the language that appear in agent coordination; Second, we give sufficient and necessary conditions of valid abstraction, based on which an efficient algorithm is developed to generate languages that are complete and optimal. Third, we apply the algorithm to dozens of environments and compare the advantages of languages generated by our algorithm with other similar languages in a path-planning domain.

The rest of the paper is organized as follows. We review the related work in Section 2. Section 3 introduces the background of multi-agent planning and presents the problem formulation of language generation. Section 4 provides the conditions of valid abstraction and describes a language generation algorithm. Section 5 implements the algorithm and evaluates the languages. Conclusions and future work are given in Section 6.

## 2. Related Work

Communication is one of the most basic and important issues in multi-agent coordination. Depending on how information is obtained, communication can be divided into implicit mechanisms, for example, pheromone [12], in which agents acquire information about their teammates through the world, and explicit mechanisms, in which agents directly transmit information through media, such as spoken languages. Explicit communication is often used in intentionally multi-agent systems, since it is efficient to share knowledge between agents [13]. In addition, a taxonomy of coordination models for mobile agent is proposed for internet applications. Based on the degrees of spatial and temporal coupling, coordination-mode is categorized into four kinds: direct, meeting-oriented, blackboard-based, and Linda-like [14].

For the multi-agent planning problem, there is work on learning communication policies, where information exchange is treated as an explicit choice that may be unreliable and incur a cost, and several approximation techniques are developed to solve the optimization problem [15,16]. To overcome the high computational complexity of solving the decentralized multi-agent decision-making problem under uncertainty, there also exists work on developing efficient online planning with selective communication algorithms [17,18]. Furthermore, a neural model trained via backpropagation is proposed to enable cooperating agents to learn to represent the information they observed and communicate it with other agents [19].

ACL (Agent communication language) is an important standard language for the communication of agents and has been well-studied in software-agent systems [20]. FIPA [21] and KQML [22] are the most popular languages, and both are constructed based on speech act theory. ACLs are composed of shared communication syntax and lexicon that are defined by a human group for different purposes. In the multi-agent cooperative control problem, information exchange between agents is often required to achieve expected goals, such as consensus [23], formation control [8], and flocking [24]. Generally, communication languages used in these methods are given in advance, and no attention is paid to the generation of the languages.

Communication in natural languages is the most obvious manner when multi-agent systems include human partners. A representative approach is the inverse semantics model [10], by which the agent with a breakdown can ask humans for help using natural languages. Recent work has explored the interactive task learning in which humans teach robots to perform tasks through natural languages [25,26]. Normally, the human-robot/agent interaction requires the agents to be able to understand natural language sentences [27], and to generate sentences representing their internal information [28]. This process requires a lot of effort and is error-prone.

Our language generation problem has a connection with the work on the origin and evolution of natural languages, which has been studied earlier in evolutionary and computational linguistics [29,30]. Most of the prior works studied language evolution in the context of evolutionary games [31], for example, the talking heads experiment [32]. In particular, a study [33] suggests a relation between natural languages and a hidden planning language that preexists in the mind. While our work addresses a similar problem, we take a step further by formulating the language generation problem in the computational model of rationality. It is worth noting that language evolution is quite a complicated problem, and many researchers from different disciplines continue to develop it. In this work, we study the language generation problem in a limiting cooperative multi-agent planning setting and hope that our work could shed some light on it.

We studied the language generation problem for planning agents from different perspectives. The general idea and framework for language construction based on state abstraction are introduced in work [34]. In this paper, we extend the formulation to a multi-agent setting and present a different kind of language. Additionally, the work [35] studies the minimal language generation problem for optimal planning. However, the language is constructed and abstracted based on predefined perception symbols. The language in this work is naturally generated and only depends on planning domains.

### 3. Problem Formulation

#### 3.1. Multi-Agent Planning Model

For simplicity, we consider a planning setting including multiple agents which act with complete information and deterministic actions. The multi-agent planning problem can be modeled as  $M = (O, A, I, G)$ , which is an extension of the STRIPS model [36], where  $O$  is a finite set of propositional variables and  $A$  is the set of joint actions of agents. Each action  $a \in A$  is given by preconditions,  $Pre(a) \subseteq O$ , add effects,  $Add(a) \subseteq O$ , and delete effects,  $Del(a) \subseteq O$ . The task of a planning domain is specified by  $(I, G)$ , where  $I \subseteq O$  and  $G \subseteq O$  denote the set of initial states and goal states, respectively.

Given a planning problem, a solution or plan is to find a sequence of states connected by agents' actions that lead the initial state to a goal state. Generally, the cost of actions is measured by a cost function. A solution is optimal if it takes the lowest action cost.

#### 3.2. Assumptions

In this work, we make several assumptions that are shown as follows:

- Agents are cooperative so that they could carry out a task that cannot be done by a single agent, and they are rational and perform the task at the least cost;
- Observation and communication are only available before task execution, and agents perform the task synchronously at each step. All agents understand the constructed language.

#### 3.3. Language Construction

**Definition 1.** (Required Coordination, RC): For a multi-agent planning problem, RC is present in a situation when there are several feasible plans that incur a conflict.

For example, two agents,  $agent_1$  and  $agent_2$ , are assigned to do a task for which there are only two optimal plans,  $p_A$  and  $p_B$ . Without communication,  $agent_1$  may act following plan  $p_A$ , while  $agent_2$  acts according to plan  $p_B$ . As a result, they execute the task sub-optimally or in failure.

**Definition 2.** (Language): Given the state set  $S$  of a multi-agent planning domain  $M$ , where  $s \in S$  is the joint state of agents, a language for the domain is a tuple  $L = (W, R)$ , where  $W$  represents a set of words, and  $R$  is the concatenation operator, denoted as “#”, which can be used to combine words. Each word  $w \in W$  denotes an abstraction of joint states and is a subset of  $S$ .

A sentence is defined as a sequence of words combined by operators. In this paper, we only consider the concatenation operator, so the sentence including  $n$  words has the following form:

$$\text{Word}_1\#\text{Word}_2\#\dots\#\text{Word}_n$$

For planning agents, to avoid potential conflicts or improve team performance, the sentence as a communicative message is considered to be a constraint on agents' actions. Since agents are rational, the receiver agents choose an optimal plan that is consistent with the communicated sentence. Please note that in this work, plans specified by a sentence refer to the optimal plans which satisfy the constraint of the sentence, and we mean a joint optimal plan when mentioning a plan.

We give an intuitive example to explain the language defined above. For a multi-agent task instance  $(S_i, S_g)$ , there are three optimal plans, shown as follows:

$$P_1 : S_i, S_{11}, S_{12}, S_{13}, S_{14}, S_g$$

$$P_2 : S_i, S_{21}, S_{22}, S_{23}, S_{24}, S_g$$

$$P_3 : S_i, S_{31}, S_{32}, S_{33}, S_{34}, S_g$$

where there is RC between  $P_1$  and  $P_2$  or  $P_3$ , and no RC between  $P_2$  and  $P_3$ . One of the valid languages is  $L = \{W_1 = (S_{14}, S_{22}, S_{31}), W_2 = (S_{11}, S_{23}, S_{33})\}$ . For instance, sentence  $W_2\#W_1$  means that agents must be in a grounding state of  $W_2$  at some moment and be in a grounding state of  $W_1$  at a later moment in task execution. Thus, the state sequence of plans specified by  $W_2\#W_1$  has the following form:

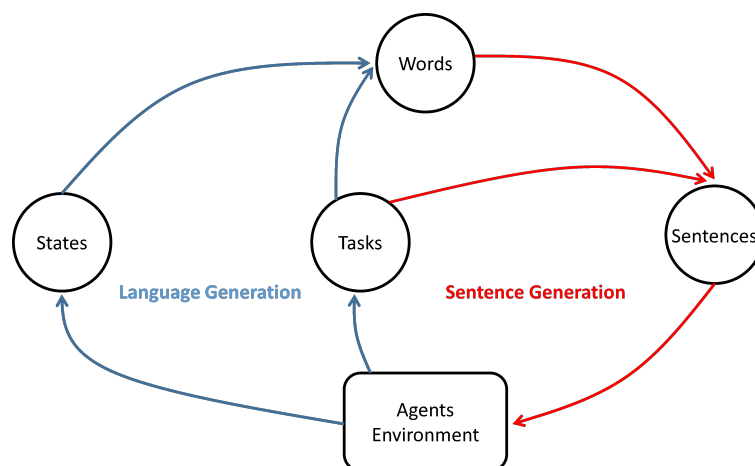
$$S_i, \dots, S_{11} \text{ or } S_{23} \text{ or } S_{33}, \dots, S_{14} \text{ or } S_{22} \text{ or } S_{31}, \dots, S_g$$

Symbol “...” denotes an omitted state sequence that can be void. Here, only  $P_1$  is consistent with the state sequence, i.e.,  $W_2\#W_1$  can specify  $P_1$ . Similarly, sentence  $W_1\#W_2$  can specify a plan set including  $P_2$  and  $P_3$ .

**Definition 3.** (Language Generation Problem, LGP): Given a multi-agent planning model  $M$ , we must find a language  $L$  that can be used to resolve the RC for any given task.

**Definition 4.** (Sentence Generation Problem, SGP): Given a multi-agent planning model  $M$ , a language  $L$ , and a task, we must find a sentence that can specify a plan set without RC.

Specifically, the LGP is about how to construct a language, and the SGP is about how to use the language to remove RC. As shown in Figure 1, words are constructed based on the states and tasks of the environment where agents act. This optimization process is operated offline. After that, given the words and tasks, sentences are generated online by agents to communicate with their teammates.



**Figure 1.** Problem formulation diagram. Blue segments: language generation process; Red segments: sentence generation process.

Next, we look at a few language properties that may be required.

*Optimality:* A language is optimal if plans specified by it are also optimal.

*Completeness:* A language is complete if it can specify a plan set that includes any plan.

*Minimality:* A language is minimal if the number of words is the smallest.

*Globality:* A language is global if it can describe the whole plan sequence.

*Locality:* A language is local if it only expresses partial specification of plan sequences.

In our prior work, the generation problem of complete global languages [34], and minimal optimal languages [35] are studied. In this work, we generate languages that are complete and local. In Section 5, we compare the performances of the languages.

The goal of this paper is to construct languages that could help agents to optimally accomplish tasks. In detail, we try to design an efficient algorithm to obtain valid state abstractions for multi-agent planning domains.

## 4. Approach

We know that a language is related to an abstraction of states. A language is useful only when it can distinguish the plans for which RC exists. In this section, we introduce the language generation and communication processes. First, the sufficient and necessary conditions for valid abstraction are provided with the proofs. Second, we propose an efficient algorithm to obtain such abstractions. Finally, we introduce the coordination process of languages.

### 4.1. Conditions

**Theorem 1.** *The state abstraction for a given domain is valid if and only if it satisfies that: For any task where RC is presented between plans  $p_1$  and  $p_2$ ,  $b_1$  and  $b_2$ , separately, is the abstracted plan sequence of  $p_1$  and  $p_2$ . Then, the following four conditions must be true: (1)  $b_1$  and  $b_2$  are not void; (2)  $b_1$  is not equal to  $b_2$ ; (3)  $b_1$  is not the partial sequence of  $b_2$ ; (4)  $b_2$  is not the partial sequence of  $b_1$ .*

**Proof of Theorem 1.** Sufficient condition: By the definition of sentences, we can see that the two plans set specified by sentence  $b_1$  and  $b_2$  have no elements in common when the four above conditions are true, i.e.,  $b_1$  and  $b_2$  can specify the plans that do not introduce RC. Necessary condition: If  $b_1$  and  $b_2$  can separately specify  $p_1$  and  $p_2$ , we have that  $b_1$  and  $b_2$  are two different sentences and do not separately express  $p_2$  and  $p_1$ . By using reduction to absurdity, if  $b_1(b_2)$  can express  $p_2(p_1)$ ,  $b_1(b_2)$  must be equal or a partial specification of the sentence sequence  $b_2(b_1)$  of  $p_2(p_1)$ . Therefore, we can reduce the conclusions.  $\square$

From the theorem, if we want to construct a useful language, we must ensure that the above conditions are satisfied during the language generation.

### 4.2. General Idea

Since the language is used to specify all optimal plans for RC tasks, it should be able to describe every plan sequence. Therefore, we first seek a smaller state set that could distinguish all plan pairs in which RC is present. Then, we generate a language by abstracting the states of the set while ensuring the conditions in Theorem 1 are true.

### 4.3. Algorithm

The language generation process includes four procedures: finding plan pairs; finding state set; finding state pairs; word generation.

*Finding plan pairs:* For each task  $t$ , we find the optimal plans by a modified  $A^*$  algorithm. The standard  $A^*$  algorithm [37] stops to search nodes when the minimum estimate of the cost value of explored nodes is equal to or greater than the value of the goal. The modified  $A^*$  algorithm continues

the search process until the value of nodes is greater than the value of the goal. For any two plans, we put them into plan pairs set  $P_s$  if RC is present.

**Finding state set:** With  $P_s$ , we need to find a state set  $S_m$  that could specify any plan pair of  $P_s$ . To make it true, a different state should at least appear in the two plan sequences. Here, we do not intend to find the minimal language, so we only need to get a state set that is approximately the smallest. First, we set state set  $S_m$  as void. Then, for each plan pair  $(x, y)$  in  $P_s$ , we remove their common states. For each plan, if no state in its sequence is the element of  $S_m$ , we add the first state of the plan into  $S_m$ . Otherwise, we address the next plan pair.

**Finding state pairs:** State pairs denote that the two states cannot be abstracted as the same word. First, we set state pairs set  $C_s$  as void. For each plan pairs  $(x, y)$  in  $P_s$ , we remove the states that do not appear in  $S_m$  and obtain the reduced plan sequences. If the two plan sequences have different lengths, we cut the longer plan into several subsequences, the length of which equals that of the shorter plan. Then, we check to see whether there are two states that are in the same place of plan sequences and appear in  $C_s$ . If they do not, we add the first state pair into  $C_s$ .

**Word generation:** Under the restriction of the state pairs of  $C_s$ , a greedy CSP (Constraint Satisfaction Problem) solver is used to assign the states of  $S_m$  to several state sets in which any pair of states does not appear in  $C_s$ . We define each state set as a word, and a language is then constructed.

The pseudo-code of the algorithm is presented in Algorithm 1.

---

**Algorithm 1** Language Generation Process
 

---

```

1: Input: Domain  $M$ ; Tasks  $\{I, G\}$ .
2: Output: Word set  $W$ .
3: Initialization: Plan pairs set  $P_s \leftarrow \{\}$ ; State set  $S_m \leftarrow \{\}$ ; Equal sequence pairs set  $E_s$ ; State pairs set  $C_s \leftarrow \{\}$ ; Conflicting state set of words  $F \leftarrow \{\}$ ; Word set  $W \leftarrow \{\}$ .
4: procedure FINDING PLAN PAIRS
5:   for task  $t \in \{I, G\}$ 
6:     Get optimal plans  $P$  of  $t$ 
7:     if  $P(i)$  and  $P(j)$  introduce RC then Put  $(P(i), P(j))$  in  $P_s$ 
8: procedure FINDING STATE SET
9:   for plan pairs  $(x, y) \in P_s$ 
10:    Remove common states of plan  $x$  and  $y$ 
11:    if not all states of  $x$  or  $y$  appear in  $S_m$  then Put a state of  $x$  or  $y$  in  $S_m$ 
12:    Get reduced sequence  $R(x), R(y)$  of  $x, y$ 
13:    if  $R(x)$  equals  $R(y)$  then Put  $(R(x), R(y))$  in  $E_s$ 
14:    elseif  $R(x)$  is longer than  $R(y)$  then
15:      Get the subsequence  $SR(x)$  whose length equals  $R(y)$ ; Put  $(SR(x), R(y))$  in  $E_s$ 
16: procedure FINDING STATE PAIRS
17:   for sequence pairs  $(e_1, e_2) \in E_s$ 
18:     for each step  $i \in |e_1|$ 
19:       if state pairs  $(e_1(i), e_2(i)) \in C_s$  then break
20:       elseif  $i = |e_1|$  then Put  $(e_1(i), e_2(i))$  in  $C_s$ 
21: procedure WORD GENERATION
22:   for state  $s \in S_m$ 
23:     for word  $w \in W$ 
24:       if  $s$  is not a member of  $F(w)$  then Abstract  $s$  as  $w$ ; Add  $s_c$  into  $F(w)$ ,  $(s, s_c) \in C_s$ 
25:       elseif  $w = W(|W|)$  then Abstract  $s$  as  $w_n$ , put  $w_n$  in  $W$ 
  
```

---

**Theorem 2.** Given a multi-agent planning domain, the languages generated by the algorithm are optimal and complete.

**Proof of Theorem 2.** For any RC plan pair,  $p_1$  and  $p_2$ , we assume that the two abstracted sentence sequences are  $b_1$  and  $b_2$ , respectively. From the language generation process,  $b_1$  and  $b_2$  are obviously not void. If the length of  $b_1$  equals the length of  $b_2$ , we know that the two first words of the sentences are



different, so  $b_1$  does not equal  $b_2$ . If they do not, the shorter sentence does not equal any subsequence of the longer sentence, so  $b_1$  does not contain or is involved in  $b_2$ . Therefore, the four conditions are always satisfied. More specifically, the plan set specified by  $b_1$  must include  $p_1$  and not include  $p_2$ , and vice versa, i.e., the languages are complete and, since the languages are generated upon optimal plans, the languages are also optimal.  $\square$

Please note that we do not consider the semantics of words in this work. However, some features representing the relationship between agents and environment can be defined in accordance with the application requirement of languages, such as the distance between the agents, and whether agents are close to the goals or not. Thus, the word function that describes the mapping from states to words, can be achieved by classical clustering methods (e.g., CLARANS algorithm [38]).

#### 4.4. Language Communication

The coordination process between agents using our language can be described as follows:

First, an optimal plan is found by a coordinator for current task. Depending on the framework of the multi-agent system, the coordinator could be an agent or a control station.

Second, the sentence of the plan can be generated based on the language, and is then sent to other agents as coordination information. In the sentence generation, the states are abstracted as the words that they belong to. It can be seen that there always exists a sentence that could express the plan.

Third, receiver agents choose their actions under the constraints of the sentence. Consequently, the task is finished without conflicts between the agents. Communication is not required when no sentence is generated, and agents can act freely.

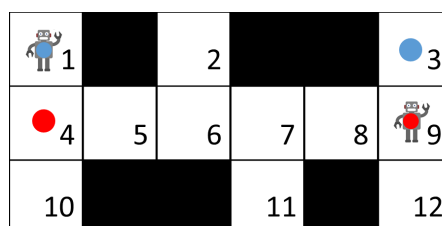
Please note that this work mainly focuses on generating coordination messages rather than obtaining task solutions. The automated planning methods [39] can be applied to find agents' plan based on the sentences.

## 5. Experiments and Results

In this section, we make several simulation experiments to verify the performance of the algorithm and the advantages of the languages in a grid-world domain. First, a simple navigation example is provided to illustrate the language generation and coordination. Furthermore, we compare our languages with the languages in [34,35] from several aspects. In the end, we implement the algorithm to more scenarios with different settings.

### 5.1. Coordination Example

A path-planning problem in a grid-world domain involving two agents,  $agent_b$  and  $agent_r$ , is shown in Figure 2. The numbered white cells are reachable for the agents and the black cells are the obstacles. In each step, the agent can move to an adjacent cell or remain where it is. Agents are not allowed to stay in the same cell or move to each other's place at the same step. Given target points, the goal of the agents is to arrive at the points with the least time and energy cost. As mentioned before, we assume that observation and communication are only available during the planning phase.



**Figure 2.** Navigation example. A Required Coordination (RC) task is that  $agent_b$  moves from point 1 to 3, and  $agent_r$  moves from point 9 to 4.

We apply the algorithm to the environment with different RC tasks. The size of the language and the computation time of the algorithm, along with the increase of the number of RC tasks, are denoted by the blue and red curves in Figure 3, respectively.

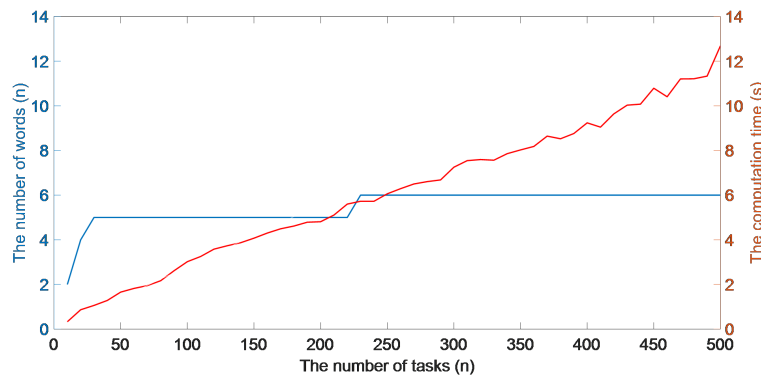


Figure 3. Language size and algorithm computation time in different task settings.

We observe that the number of language words does not always increase as more RC tasks are included, since the generated language could apply to other RC tasks. The language of the environment involving all RC tasks is  $L = \{W_1, W_2, W_3, W_4, W_5, W_6\}$ . The states of each word are shown in Table 1.

Table 1. Generated language for the example.

The Words in L	The States in W
$W_1$	$S_{4,1}, S_{7,8}$
$W_2$	$S_{4,5}, S_{7,6}$
$W_3$	$S_{4,10}, S_{7,11}$
$W_4$	$S_{6,5}, S_{9,12}$
$W_5$	$S_{6,2}, S_{9,3}$
$W_6$	$S_{6,7}, S_{9,8}$

Where state  $S_{x,y}$  indicates that  $agent_b$  is at point  $x$  and  $agent_r$  is at point  $y$ . For a planning instance, the task of  $agent_b$  and  $agent_r$  is moving from 1 and 9 to 3 and 4. Assume that every time step and movement take a cost of 1, respectively. Thus, there are seven optimal plans for the two agents, which are shown as follows:

$P_1 : S_{1,9}, S_{1,8}, S_{4,7}, S_{5,6}, S_{6,2}, S_{7,6}, S_{8,5}, S_{9,4}, S_{3,4};$	$B_1 : W_5 \# W_2$
$P_2 : S_{1,9}, S_{4,8}, S_{4,7}, S_{5,6}, S_{6,2}, S_{7,6}, S_{8,5}, S_{9,4}, S_{3,4};$	$B_2 : W_5 \# W_2$
$P_3 : S_{1,9}, S_{4,8}, S_{5,7}, S_{5,6}, S_{6,2}, S_{7,6}, S_{8,5}, S_{9,4}, S_{3,4};$	$B_3 : W_5 \# W_2$
$P_4 : S_{1,9}, S_{4,9}, S_{5,8}, S_{6,7}, S_{7,11}, S_{8,7}, S_{9,6}, S_{3,5}, S_{3,4};$	$B_4 : W_6 \# W_3$
$P_5 : S_{1,9}, S_{4,8}, S_{5,8}, S_{6,7}, S_{7,11}, S_{8,7}, S_{9,6}, S_{3,5}, S_{3,4};$	$B_5 : W_6 \# W_3$
$P_6 : S_{1,9}, S_{4,8}, S_{5,7}, S_{6,7}, S_{7,11}, S_{8,7}, S_{9,6}, S_{3,5}, S_{3,4};$	$B_6 : W_6 \# W_3$
$P_7 : S_{1,9}, S_{4,8}, S_{5,7}, S_{6,11}, S_{7,11}, S_{8,7}, S_{9,6}, S_{3,5}, S_{3,4};$	$B_7 : W_3$

In this case, the agents may choose any plan to follow if they do not communicate. However, they conflict with each other in task execution if an agent chooses anyone from the first three plans and its partner chooses another plan from the last four plans. In other words, RC is present for the task. Based on the language  $L$ , we generate the sentences of the seven plans, which are shown following the plans. Sentence  $W_5 \# W_2$  can express agents' preference for plan, which specifies a no RC plan set including  $P_1$ ,  $P_2$ , and  $P_3$ . Similarly, sentence  $W_6 \# W_3$  can specify a plan set that includes  $P_4$ ,  $P_5$ , and  $P_6$ , and sentence  $W_3$  can specify  $P_4$ ,  $P_5$ ,  $P_6$ , and  $P_7$ . Thus, the potential conflicts between the agents can be solved by communicating one of the sentences.



## 5.2. Languages Comparison

To measure the performance of the languages constructed in this work, we compare our languages with languages generated by the method in [24], denoted as CGL (complete global language), and the method in [25], denoted as MOL (minimal optimal language), in terms of message lengths, specified plans, and computation time. Figure 4 is the test example in which there are 10 task points, marked as blue stars, which need to be continually visited by two agents.

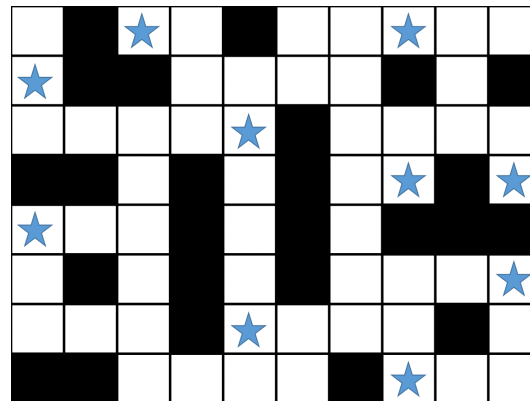


Figure 4. Test example.

For this environment, there are a total of 8100 tasks (a task is specified by a pair of agents' states  $\langle \text{initial}, \text{goal} \rangle$ ,  $8100 = 10 * 9 * 10 * 9$ ), of which 350 tasks introduce RC. We obtain that the number of language words generated by our approach, CGL, and MOL are 6, 7, and 2, respectively. For the sake of contrast, we conduct 100 RC tasks at random and record the relevant data. Figures 5–7 separately show the length of the communicative messages, the number of specified optimal plans, and the time cost of sentence generation, respectively. The red, purple, and green curves represent the results achieved by our method, CGL, and MOL, respectively. In Figure 5, the blue curve denotes the number of states in the plans. As we can see, coordination messages generated by our algorithm are quietly shorter than those of other communication methods, which greatly reduces the burden of communication of agents. Figure 6 shows that our sentences could specify many available plans for agents to follow, which gives them more flexibility to execute tasks. Although the constructed language in this work is not minimal, our approach has nearly comparable benefits to those of MOL in this respect. Furthermore, as can be seen in Figure 7, the computation costs of language planning using our method are much less than the costs of MOL. As with CGL, the complexity of the SGP in our method is almost equivalent to the task planning problem. In general, the results show that our languages offer considerable advantages compared to CGL and MOL in these criteria.

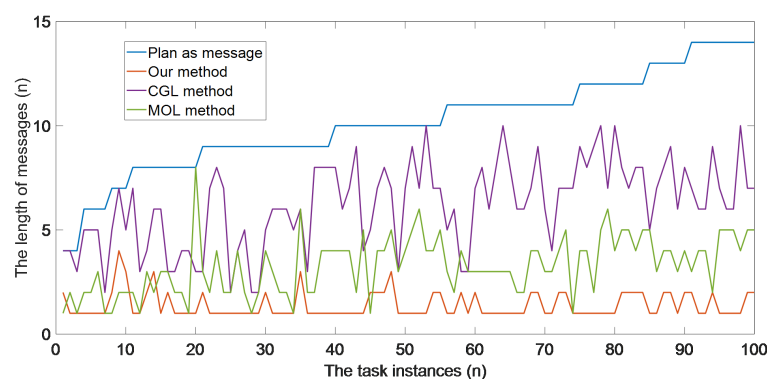


Figure 5. Length of plans and sentences.

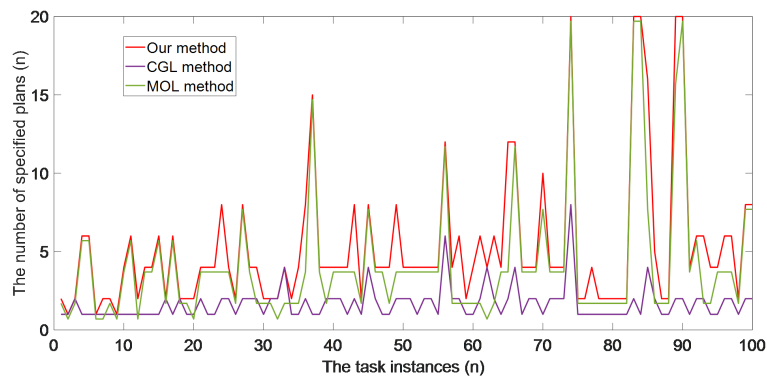


Figure 6. Number of plans specified by sentences.

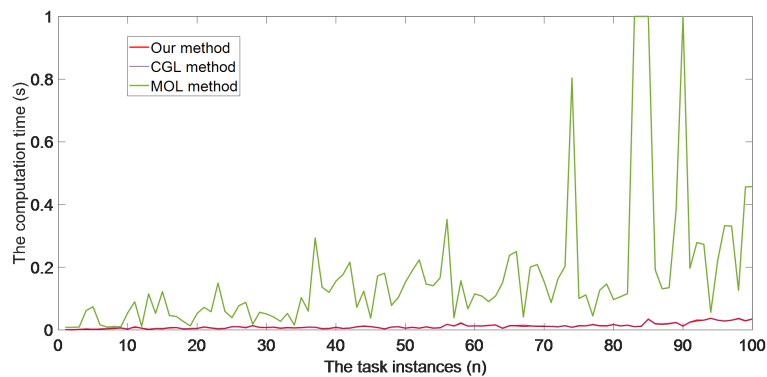


Figure 7. Computation cost of sentences generation.

### 5.3. Different Scenarios

We also implement our algorithm to the path-planning problem in 30 grid-world scenarios. For each scenario, we separately run 500 RC tasks in the setting of two, three, and four agents. Figures 8 and 9 demonstrate the number of agents' joint states and our language words used in these tasks, respectively. The number of words is far smaller than the states, which means that agents require less effort to understand the messages by our approach. To assess the benefits of the languages in terms of communication cost and behavioral flexibility of agents, we compute their coordination sentences and specified plans for all task plans. Bars in Figure 10 represent the average decrement of the length of sentences compared with plan sequences. The communication costs decline by 60% on average using the languages. Bars in Figure 11 indicate the average amount of plans specified by the sentences. From these results, we can learn that the constructed languages are quite useful for multi-agent coordination.

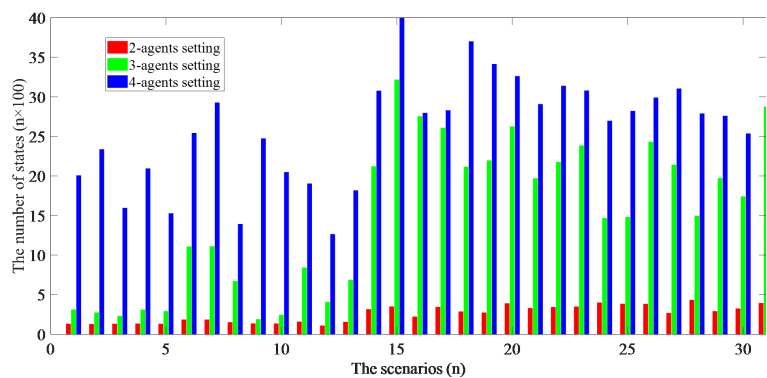


Figure 8. Number of states involved in each scenario.

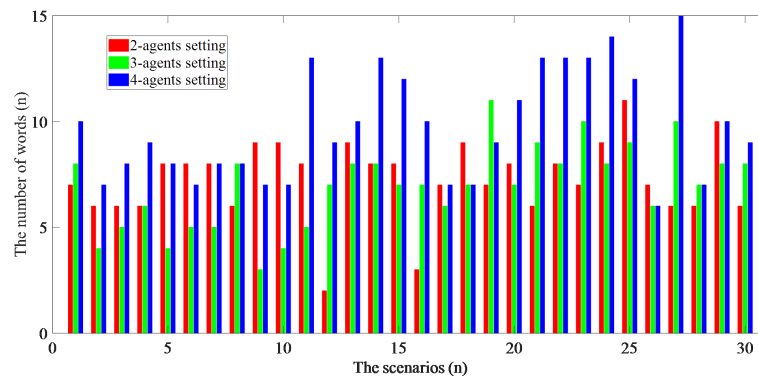


Figure 9. Number of generated words for each scenario.

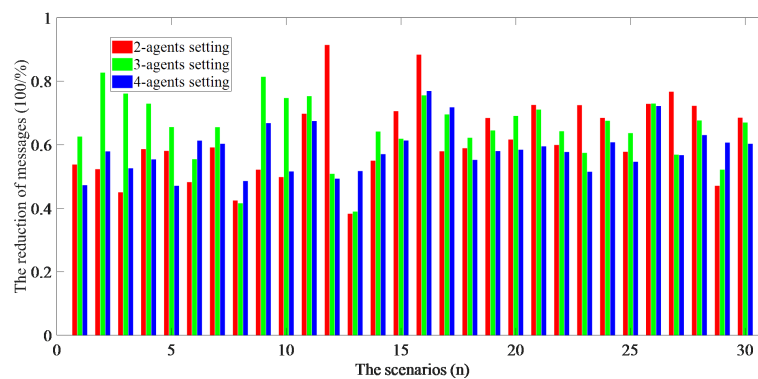


Figure 10. Average decrement of communication costs.

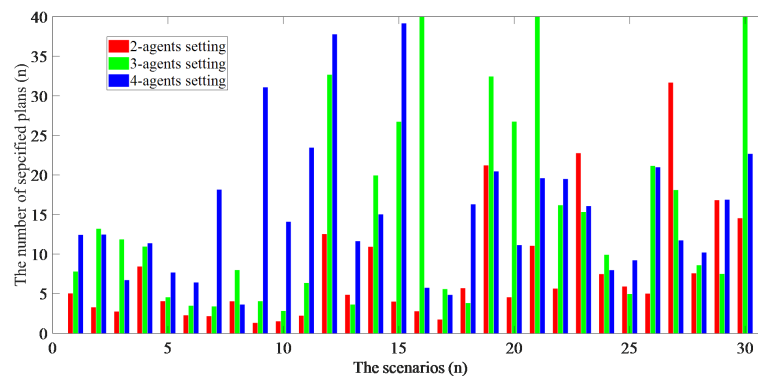


Figure 11. Average number of plans specified by sentences.

It should be noted that the classical planning problem is known to be PSPACE-complete [40]. The LGP is considered to be more difficult, and finding a minimal language is PSPACE-hard [34]. It is challenging to solve the LGP involving more agents. In this work, we only apply our method to the path-planning problem with a small number of agents, and a more efficient method remains to be further exploited when the problem increases. Fortunately, the LGP can be addressed offline in a centralized manner. Generating a communication sentence for agents' coordination is rather simpler than language generation, which is not harder than finding an optimal plan.

## 6. Conclusions

In this paper, we provided a new coordination approach using simple languages for multi-agent systems. The languages are not manually defined and are naturally generated via state abstraction. The benefits of the languages include specifying optimal plans, empowering agents more flexibility in behaviors, and reducing communication costs. The experiment results confirmed that efficient

languages for task execution can be constructed based on agents' own internal representation in the computational model of rationality. They also suggest that the LGP in multi-agent planning domains may provide a perspective for creating a "natural language" for autonomous agents.

In future work, we intend to investigate the LGP involving more agents. There are multiple ways in which scalability can be improved. A method is to introduce communication between the agents during plan execution, which breaks plans into plan segments, essentially reducing the number of planning problems. A second method is to study the strategy for pairwise coordination between the agents using the language constructed for two agents. Another interesting direction is to construct languages that work in varying environments. A possible solution is dividing a new environment into subspaces that are isomorphic to the original environment where the language is constructed, and augmenting it when this cannot be done.

**Author Contributions:** Conceptualization, L.W. and Q.G.; methodology, L.W.; software, L.W.; formal analysis, L.W. and Q.G.; writing—original draft preparation, L.W. and Q.G. All authors reviewed the results and approved the final version of the manuscript.

**Funding:** This research received no external funding.

**Acknowledgments:** The authors thank the anonymous reviewers for their thoughtful comments on this paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this paper:

RC	Required Coordination
LGP	Language Generation Problem
SGP	Sentence Generation Problem
A	Actions
S	States
P	Plans
L	Languages
W	Words
B	Sentences
CGL	Complete Global Language
MOL	Minimal Optimal Language

## References

1. Wooldridge, M. *An Introduction to Multiagent Systems*, 2nd ed.; Wiley: Torquay, UK, 2009.
2. Farinelli, A.; Boscolo, N.; Zanutto, E.; Pagello, E. Advanced approaches for multi-robot coordination in logistic scenarios. *Robot. Auton. Syst.* **2017**, *90*, 34–44. [[CrossRef](#)]
3. Werfel, J.; Petersen, K.; Nagpal, R. Designing collective behavior in a termite-inspired robot construction team. *Science* **2014**, *343*, 754–758. [[CrossRef](#)] [[PubMed](#)]
4. Bernard, M.; Kondak, K.; Maza, I.; Ollero, A. Autonomous transportation and deployment with aerial robots for search and rescue missions. *J. Field Robot.* **2011**, *28*, 914–931. [[CrossRef](#)]
5. Wurman, P.R.; D'Andrea, R.; Mountz, M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Mag.* **2008**, *29*, 9–20.
6. Jordán, J.; Palanca, J.; Val, E.D.; Julian, V.; Botti, V. A multi-agent system for the dynamic emplacement of electric vehicle charging stations. *Appl. Sci.* **2018**, *8*, 313. [[CrossRef](#)]
7. Reynolds, C.W. Interaction with a group of autonomous characters. In Proceedings of the Game Developers Conference, San Jose, CA, USA, 15–19 March 2000; pp. 449–460.
8. Dong, X.; Yu, B.; Shi, Z.; Zhong, Y. Time-varying formation control for unmanned aerial vehicles: Theories and applications. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 340–348. [[CrossRef](#)]
9. Torreño, A.; Onaindia, E.; Sapena, O. FMAP: Distributed cooperative multi-agent planning. *Appl. Intell.* **2014**, *41*, 606–626. [[CrossRef](#)]

10. Tellex, S.; Knepper, R.; Li, A.; Rus, D.; Roy, N. Asking for help using inverse semantics. In Proceedings of the RSS, Rome, Italy, 13–17 July 2014.
11. Gong, Z.; Zhang, Y. Temporal spatial inverse semantics for robots communicating with humans. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 4451–4458.
12. Almeida, J.D.; Nakashima, R.T.; Neves, F., Jr.; Valéria, L.; de Arruda, R. Bio-inspired on-line path planner for cooperative exploration of unknown environment by a Multi-Robot System. *Robot. Auton. Syst.* **2019**, *112*, 32–48. [[CrossRef](#)]
13. Parker, L.E. Multiple mobile robot systems. In *Springer Handbook of Robotics*; Springer: Berlin, Germany, 2016; pp. 921–941.
14. Cabri, G.; Leonardi, L.; Zambonelli, F. Mobile-agent coordination models for internet applications. *Computer* **2000**, *33*, 82–89. [[CrossRef](#)]
15. Goldman, C.V.; Zilberstein, S. Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis. *arXiv* **2011**, arXiv:1107.0047.
16. Goldman, C.V.; Allen, M.; Zilberstein, S. Learning to communicate in a decentralized environment. *Auton. Agents Multi-Agent Syst.* **2007**, *15*, 47–90. [[CrossRef](#)]
17. Wu, F.; Zilberstein, S.; Chen, X. Online planning for multi-agent systems with bounded communication. *Artif. Intell.* **2011**, *175*, 487–511. [[CrossRef](#)]
18. Roth, M.; Simmons, R.G.; Veloso, M.M. Reasoning about joint beliefs for execution-time communication decisions. In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht, The Netherlands, 25–29 July 2005; pp. 786–793.
19. Sukhbaatar, S.; Fergus, R. Learning multiagent communication with backpropagation. *Proc. NIPS* **2016**, 2244–2252.
20. Soon, G.K.; On, C.K.; Anthony, P.; Hamdan, A.R. A review on agent communication language. In *Computational Science and Technology*; Springer: Singapore, 2019; pp. 481–491.
21. Sadek, M.D.; Bretier, P.; Panaget, F. ARTIMIS: Natural dialogue meets rational agency. In Proceedings of the IJCAI, Nagoya, Japan, 23–29 August 1997; pp. 1030–1035.
22. Finin, T.; Fritzson, R.; McKay, D.; McEntire, R. KQML as an agent communication language. In Proceedings of the Third International Conference on Information and Knowledge Management, Gaithersburg, MD, USA, 29 November–2 December 1994; pp. 456–463.
23. Nowzari, C.; Garcia, E.; Cortés, G. Event-triggered communication and control of networked systems for multi-agent consensus. *Automatica* **2019**, *105*, 1–27. [[CrossRef](#)]
24. Olfati-Saber, R. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. Autom. Control* **2006**, *51*, 401–420. [[CrossRef](#)]
25. Rybski, P.E.; Yoon, K.; Stolarz, J.; Veloso, M.M. Interactive robot task training through dialog and demonstration. In Proceedings of the 2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI), Arlington, VA, USA, 9–11 March 2007; pp. 49–56.
26. Chai, J.Y.; Gao, Q.; She, L.; Yang, S.; Saba-Sadiya, S.; Xu, G. Language to action: Towards interactive task learning with physical agents. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 2–9.
27. Paul, R.; Arkin, J.; Roy, N.; Howard, T.M. Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators. In Proceedings of the 2016 Robotics: Science and Systems, Ann Arbor, MI, USA, 18–22 June 2016.
28. Fang, R.; Doering, M.; Chai, J.Y. Collaborative models for referring expression generation in situated dialogue. In Proceedings of the AAAI Conference on Artificial Intelligence 2014, Québec City, QC, Canada, 27–31 July 2014; pp. 1544–1550.
29. Steels, L. Evolving grounded communication for robots. *Trends. Cogn. Sci.* **2003**, *7*, 308–312. [[CrossRef](#)]
30. Spike, M.; Stadle, K.; Kirby, S.; Smith, K. Minimal requirements for the emergence of learned signaling. *Cognit. Sci.* **2017**, *41*, 623–658. [[CrossRef](#)] [[PubMed](#)]
31. Lazaridou, A.; Peysakhovich, A.; Baroni, M. Multi-agent cooperation and the emergence of (natural) language. *arXiv* **2016**, arXiv:1612.07182.
32. Steels, L. *The Talking Heads Experiment: Origins of Words and Meanings*; Language Science Press: Berlin, Germany, 2015.

33. Steedman, M. Plans, affordances, and combinatory grammar. *Linguist. Philos.* **2002**, *25*, 723–753. [[CrossRef](#)]
34. Zhang, Y.; Wang, L. From abstractions to “natural languages” for planning agents. *arXiv* **2019**, arXiv:1905.00517.
35. Wang, L.; Guo, Q. Generating the minimal optimal language for cooperative agents. *IEEE Access* **2019**, *7*, 60348–60358. [[CrossRef](#)]
36. Fikes, R.E.; Nilsson, N.J. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* **1971**, *2*, 189–208. [[CrossRef](#)]
37. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [[CrossRef](#)]
38. Ng, R.; Han, J. Clarans: A method for clustering objects for spatial data mining. *IEEE Trans. Knowl. Data Eng.* **2002**, *14*, 1003–1016. [[CrossRef](#)]
39. Ghallab, M.; Nau, D.; Traverso, P. *Automated Planning and Acting*; Cambridge University Press: Cambridge, UK, 2016.
40. Bylander, T. The computational complexity of propositional STRIPS planning. *Artif. Intell.* **1994**, *69*, 165–204. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).