From Abstractions to “Natural Languages” for Coordinating Planning Agents

Yu Zhang  
Arizona State University  
Email: yzhan442@asu.edu

Li Wang  
Beijing Institute of Technology  
Email: wangli2015@bit.edu.cn

Abstract—Despite significant advancements in developing autonomous agents, communication between them often relies on a set of pre-specified symbols for a given domain. In this paper, we investigate the automatic construction of these symbols from abstractions to form “natural languages” for such agents. The focus of this initial investigation is on a task planning setting where one agent (the speaker) directly communicates a “plan sketch” to another agent (the listener) to achieve coordination. In contrast to prior work, we view language formation as a fundamental requirement for resolving miscoordination. This view enables us to “compute” a language from the ground up by mapping physical states to symbols, thus reverse engineering the function of languages. Languages that arise from this process are only approximately expressive of the actual plans, meaning that they specify abstractions over the plan space, which is not only theoretically appealing as it provides the desired flexibility to the listener to choose its plan during execution, but also practically useful since it both reduces the communication cost of the speaker and computational cost of the listener. We formulate this language construction problem and show that it is NEXP-complete. An approximate solution is then developed to relate this problem to task planning problems that have efficient off-the-shelf solutions. Finally, we discuss a multi-agent path-finding domain in our evaluation to provide a comprehensive set of results to illustrate the benefits of the constructed languages and their applications.

I. INTRODUCTION

Many real-world tasks require the coordination of multiple agents. A key consideration in facilitating such coordination is the adoption of a communication language. While humans are endowed with the gift of natural languages, autonomous agents must often rely on a set of symbols that are pre-specified by experts of a given domain [4][14][14][31]. This approach, however, is neither practical nor efficient. An alternative approach is to implement an interface layer, often in the form of a natural language generation and understanding framework [40][27], so that agents can convert between semantic meanings and natural language sentences. It comes with the additional benefit of better human interpretability and hence is desirable when there are human agents in the loop. On the other hand, it involves significant technical challenges and is itself an active research area [15][4][3][19].

In this paper, we set out to investigate the problem of automatic language construction for autonomous agents. The focus of this initial study is on a task planning setting where two agents must communicate to coordinate their behaviors before execution. In particular, one agent (the speaker) directly communicates a “plan sketch” to another agent (the listener) to achieve coordination, which captures the speaker’s preferences for the task plan. In contrast to prior work, we view language construction as a fundamental requirement for resolving miscoordination, which enables us to compute a language from the ground up by mapping physical states to symbols, thus reverse engineering the function of languages. In particular, our language construction process is driven by the fundamental question of when agents are required to coordinate. It is observed that the presence of coordination conflicts introduces the need for communication and, in turn, languages. This effectively converts the language construction problem to the problem of searching for a symbolic system that resolves those conflicts. More importantly, the answer to the question demarcates situations where coordination is required so that languages become a necessity, from situations where it is not essential but still useful such as for providing redundancy. Furthermore, languages that arise from this process are guaranteed to resolve coordination conflicts, and at the same time are only approximately expressive of the actual plans, meaning that they specify abstractions over the plan space. This feature is not only theoretically appealing as it provides the desired flexibility to the listener to accommodate preferences of its own while respecting those of the speaker but also practically useful since it both reduces the communication cost of the speaker and computational cost of the listener, as we will show.

Our work addresses the above question by answering specifically how a language naturally arises from a problem domain that requires coordination. First, we define the notion of required coordination (RC) to identify the situations where coordination is required. Our goal then becomes that of searching for a symbolic system that resolves these RCs. Observing that a symbolic system is nothing but an abstraction of the domain, we base our language construction method on a general temporal-state abstraction. While finding a language that resolves RC is not difficult, the challenge lies in identifying the ones that also provide more flexibility to the listener. This can be achieved by optimizing the domain abstraction and our focus here is on minimizing a language in terms of its word set, which is generally aligned with language design principles [7][8] and often represents a desirable language

1The notion of required coordination may appear similar to others discussed before [10][56], where different problems are often involved. The work of [43] is particularly relevant but the focus there has been on applications of the constructed languages without providing a formal or complete treatment.
property [35]. We show that searching for such a language is NEXP-complete. In light of the complexity, we develop an approximate solution that addresses it by solving a set of planning problems, which have efficient off-the-shelf solutions. For evaluation, we discuss a multi-agent path-finding domain to provide a complete set of results to illustrate the benefits of the constructed languages and their applications.

The contribution here is three-fold. First, we provide a new perspective on and a formal treatment of language construction for autonomous agents as an optimization problem based on the notion of required coordination. We establish its theoretical foundation in a task planning setting with two agents. Second, we provide an approximate solution to a formulation of this problem that minimizes the word set by relating it to planning problems. Third, we provide a comprehensive evaluation of the languages generated by our method in a commonly used domain to demonstrate its many benefits.

II. RELATED WORK

The emergence and evolution of languages have been studied extensively in evolutionary and computational linguistics [34, 9, 6]. Most of the work there has considered the questions in the context of evolutionary games [36], and often in an iterated learning setting [23]. Steedman’s work [33] is particularly inspiring in which he suggests a connection between natural languages and a hidden planning language that preexists in mind, although it falls short of establishing their computational connections. From this perspective, our work may be most connected to a prior study that considers the incentive of communication in a game-theoretical framework [2]. In contrast, our work is focused on a cooperative setting where languages naturally arise out of the need for coordination. While we draw inspiration from these prior works, it is our intention here to address the language construction problem from a fundamentally different computational view.

A language represents a structured symbolic system that maps symbols to semantic meanings grounded in the environment [20, 36, 39, 41, 19]. In this work, we take the path to reverse engineer this process by considering the mapping from physical states to symbols for language construction. These symbols are used to specify an abstraction over the plan space such that the expressed plans do not introduce coordination conflicts. The idea of applying abstractions to planning problems is hardly new, and has been well studied in both path planning and task planning methods [17, 22, 26, 12]. The notion of temporal-state abstraction used in our language construction process is similar to options in semi-MDP and LTL expressions in temporal logic [38, 42, 11]. However, abstractions in these prior works have mostly been discussed in the planning context to reduce the computational cost [11, 29, 24], instead of in connection with the fundamental requirement of coordination. Similarly, conflict-oriented methods have been studied in multi-agent planning problems [32, 45] but never from a language construction perspective.

Our problem may also appear similar to the task of learning or planning to communicate. Along this line, there exists prior work on developing communication schemes to maximize the team performance, where it is considered either as a planning or learning problem [17, 15, 28, 3]. For example, a communication scheme can arise from a Dec-POMDP framework where communicating actions are modeled specifically [18], which essentially augment the observation function. The set of symbols in these works are assumed to be given a priori and the focus there has been on learning [28] or optimizing [18] the association between those symbols and semantic meanings to improve a reward metric. Instead, we construct symbols from physical states from the ground up as the need for coordination arises in the given domain. In this way, the languages constructed from our method neither suffer from “overfitting” nor “underfitting” of those symbols.

III. BACKGROUND

The utility of a language is for a speaker to convey important information to the listeners. The ability to use languages is known to exist in many natural species [21], each having evolved a language of its own. In this paper, we intend to study how a language may be constructed for a set of autonomous agents from a task planning setting. Next, we present our problem settings along with the simplifications made in this work, followed by a motivating example.

A. Problem Settings

- Two rational agents in a cooperative task planning setting. Each agent knows about the other’s capabilities, the environment, and the joint initial and goal states.
- Both agents must commit to a task plan before execution.
- The plan preferences of each agent are private and may change arbitrarily. However, both agents can communicate information using a shared language to convey their
preferences to the other agent. Both agents understand about the language.

- It is always in each agent’s interest to not only respect preferences of its own but also the other’s if possible. More specifically, the speaker communicates a “plan sketch” (i.e., an abstract plan) that captures its plan preferences (as a set of plans), which are to be respected by the listener if at all possible (i.e., there exists an overlap between the agents’ preferences).

The first condition means that each agent may independently anticipate a joint plan for both agents, and only optimal plans are considered. Given a cooperative setting, the agents are tasked only to optimize the team performance. In this work, we use makespan as the team performance metric. The second condition applies to many situations when it is difficult (if not outright infeasible) to change plans after execution starts. In such a case, without communication, the two agents may anticipate different joint plans due to their individual preferences and end up with coordination conflicts (i.e., failing to reach the goal state after the execution from both agents completes). Coordination is achieved by one of the agents communicating its preferences as a path sketch to the other agent. The listener is expected to respect the speaker’s preferences if they overlap with the preferences of its own. Hence, the goal of a language in our problem settings is to resolve conflicts while providing as much flexibility as possible when providing a plan sketch in various situations.

B. Motivating Example

Consider a cooperative and distributed path-finding domain in Fig. 1 with two agents A and B with three rooms 1, 2 and 3. A task in this domain is for the agents to start from an initial state and reach a goal state, where a state specifies locations for both agents. The initial and goal states may specify any locations of the agents in the environment. Both agents share the same optimization criterion for the team (i.e., minimizing the makespan of the task). Each agent can move to an adjacent room or stay where it is in the next time step but they are not allowed to stay in the same room or switch locations at the same time step to avoid collisions. Without coordination, the agents may choose their actions arbitrarily according to any joint optimal plan that is aligned with their preferences, which can lead to coordination conflicts (see Fig. 1).

In such a case, communication is needed for the agents to coordinate their actions due to the different preferences. For example, agent A may prefer to first move and then wait while B may prefer the other way around. Such preferences may also be changed arbitrarily although we assume that they do not change within a negotiation cycle (i.e., from when the speaker sends a message to when it receives an acknowledgment from the listener about whether or not a commitment can be made). The utility of a language is to implement such communication under all tasks (i.e., all initial and goal state pairs). In this example, a language may be one that enables the agents to express something like “clockwise” or “counterclockwise” movements. This happens to be expressible by a language computed by our approximate solution to the language construction problem. As we will discuss later, the symbols in the constructed language for this example form an abstraction over the joint state. The resulting abstract states (i.e., symbols) are automatically constructed and can be used to specify the location of either agent A or B regardless of the other agent. In this way, a clockwise movement can be expressed by specifying the locations of the agent in the desired sequence (i.e., the symbols are compositional in our languages).

IV. Problem Formulation

A. Basics for Task Planning

We model the domain and environment based on the STRIPS model $M = (P.A)$, where $P$ is the set of propositional state variables, and $A$ is the set of agent joint actions. Each joint action $a \in A$ (which specifies the individual action of each agent) is associated with a set of preconditions, $pre(a) \subseteq P$, add effects, $add(a) \subseteq P$, and delete effects, $del(a) \subseteq P$. We assume that every joint action has a cost of 1 and the optimality criterion is to minimize the makespan. The agents must work together to achieve tasks in the form of state pairs $(I,G) \in S \times S$, where $S = 2^P$ is the state space, and $I$ and $G$ are the initial and goal states. Note that this deviates from the STRIPS model where $G$ can be a partial state. We adopt a more strict goal specification here to simplify the discussion. A discussion on the removal of this restriction is delayed to future work. Each such state pair introduces a planning problem $O = (P.A.I,G)$.

Given any task as a planning problem $O = (P.A.I,G)$, each agent may choose a joint plan based on its individual preferences. However, as seen in the motivating example in Fig. 1, problems may occur when this joint plan is chosen differently by each agent. In such cases, communication is required to coordinate them. The necessity of coordination is tied to the notion of required coordination, which is captured in the definition below:

**Definition 1** (Required Coordination (RC)). Given any task $(I,G)$ that results in a planning problem for a domain, a required coordination is the condition for two plans $\pi_1 \in \Pi$ and $\pi_2 \in \Pi$ such that $(\pi_1, \pi_2) \notin \Pi$ or (non-exclusive) $(\pi_1^A, \pi_2^B) \notin \Pi$, where $\pi_1 = (\pi_1^A, \pi_1^B), \pi_2 = (\pi_2^A, \pi_2^B)$ and $\Pi$ is the set of joint (optimal) plans for the task.

A plan with a superscript (e.g., $\pi^A$) denotes the individual plan (the sequence formed by individual actions) for the respective agent and $(\pi^A, \pi^B)$ combines the individual plans $\pi^A$ and $\pi^B$ back into a joint plan. Intuitively, an RC condition defines a situation for the existence of two different plans for a given task, such that a task failure will occur (i.e., goal not reached) after the execution completes had the agents chosen them differently. In such cases, we also refer to that the two plans introduce RC. Notice that two different plans do not necessarily introduce RC. See Fig. 2 for an example.

**Proposition 1.** Given a planning domain model $M = (P.A)$, no coordination is necessary when no RC is present for any
Fig. 2: Three different joint plans (with individual plans also shown) for the task in our motivating example: the first two plans introduce no RC: $π₁$ from $π₁$ (top) can be switched with its counterpart from $π₂$ without affecting the task completion. Both $π₁$ and $π₂$ however introduce RC with $π₃$.

A temporal-state constraint specifies a constraint in the form of $ζ$ = $S₁, S₂, ..., Sₙ$, where $Sᵢ ∈ 2^S$. A plan $π$ satisfies a constraint $ζ$ if and only if there exists a set of monotonically increasing positive integers $j₀, j₁, j₂, ..., jₙ$ with $j₀ = 0$ and $jₙ = |π|$ such that $π[jₙ⁻¹ + 1 : jₙ] ⊆ Sₖ$ ($n ≥ k ≥ 1$), $π[jₙ⁻₁] ∉ Sₖ$ ($n ≥ k > 1$) and $π[jₙ + 1] ∉ Sₖ$ ($n > k ≥ 1$), where $π$ is the state sequence that corresponds to $ζ$.

Brackets are used to index into $π$ and the symbol $:$ indicates a range of indices (inclusive); $π[x : y]$ thus returns a set. An illustration of a TSC is provided in Fig. 3. Alternatively, we also refer to that a TSC expresses a plan if the plan satisfies the constraint. It is worth noting that the abstraction used in TSC may be considered as a strict generalization of options in semi-MDPs [43], which specify only the initial and final states. Instead, a TSC specifies a sequence of abstract states that must be visited by a plan in order. While this distinction may not be important for planning purposes, since a TSC can be constructed from options, it could affect the construction of a language significantly since TSC allows for more flexible expressions. Similarly, LTL and CTL [42, 11] may also be used to specify TSCs. A fundamental difference is that our focus here is not on planning where the abstractions and their associated physical states are given a priori, and instead on how the abstractions naturally arise from the requirement of coordination. Next, we formally define language using TSCs.

Definition 3 (Language). A language for a domain $M = (P,A)$ is a pair $L = (W,O)$ where $W$ is the set of words and $O$ the set of compositional operators that can be applied to the words. Each word $w ∈ W$ is a TSC.

Consider the natural language word zigzag that is defined as "have or move along in a zigzag course", which specifies a temporal-state constraint on the movement of an agent. The language operators are applied to the words (which are also referred to as basic TSCs) in a language to introduce more complex TSCs (referred to as sentences). For example, a concatenation operator may be used to express "move along in a zigzag course and then go forward" when used to connect zigzag and forward. Any given language thus introduces a set of TSCs for the domain.

Not all languages are of interest to the agents. To be useful, a language must at least be able to resolve RCs. Such languages are referred to as coordination languages:

Definition 4 (Coordination Language). A language is a coordination language for a domain if the following conditions hold: given any (optimal) plan $π$ for any task, 1) a sentence of the language expresses it; 2) for any sentence of the language that expresses it, the set of all plans for the task that are expressed by the sentence must introduce no RC.

More intuitively, since a sentence may correspond to multiple plans, a coordination language ensures that the set of plans that are expressed do not introduce RC. In such a case, the flexibility of the listener to choose a plan, as long as it is constrained by a sentence that expresses it, is guaranteed to...
not introduce coordination conflicts. However, a coordination language may be created too stringently, such as when it expresses only a single plan for any task. Such languages are undesirable since it significantly restricts the listener’s flexibility. Hence, we further prefer languages that are more flexible, or approximately expressive of plans. One way to build in more flexibility is to minimize the number of words in a coordination language. We will focus our following discussion on constructing coordination languages that have a minimal word set. To simplify the formal analysis in this work, we will hold the set of operators fixed and simple (i.e., with only concatenation), and further assume state abstractions only (i.e., words are all state abstractions or \( w \in 2^3 \) for all \( w \in W \)). Discussions for removing these simplifications will be delayed to future work.

**Theorem 1.** The decision problem, \( D_{\text{MinW}} = (P, A, T, O, K) \), to determine whether a coordination language exists with word size \( |W| = K \), under \( O = \{ \cdot \} \) (concatenation) with state abstractions is NEXP-complete.

**Proof:** First, we prove that the problem can be solved by a Non-deterministic Turning Machine (NTM) in exponential time. This machine can guess the \( W \) by enumerating all possible word set with \( K \) words, which can be done in exponential time. For each candidate word set, checking whether it forms a coordination language with \( O = \{ \cdot \} \) will not use more than an exponential amount of time, which involves computing the set of optimal plans for any given initial and goal states and validating that the language resolves all RCs subject to the conditions in Def. 4.

Next, we show that the decision problem \( D_{\text{MinW}} \) can be reduced from the TILING problem, which has the form of \( D = (L, H, V, N) \) and is known to be NEXP-complete [5]. In a TILING problem, we are given a board of size \( N \times N \), a set of tile types \( L = \{ tile_0, \ldots, tile_k \} \), and a set of horizontal and vertical compatibility relations, \( H, V \subseteq L \times L \). A tiling is a mapping \( f : \{1 \ldots N\} \times \{1 \ldots N\} \rightarrow L \) and is consistent if and only if \( f(0,0) = tile_0 \) and each pair of the horizontally or vertically adjacent cells satisfies one of the horizontal or vertical compatibility relations, respectively. The TILING problem can be reduced to a \( D_{\text{MinW}} \) problem that involves a modified two-agent path-finding domain in a grid-world of size \( N \times N \). For any given task, the agents always start in the same location and must move to the same goal location while minimizing the makespan. They can move horizontally or vertically to an adjacent cell and choose a label from \( L \) to label the cell that it moves into. The location \((0,0)\) is labeled with \( tile_0 \) initially and other locations are unlabeled. A failure will be resulted when they choose a different location to move into, different labels for the same next location, or that the move and labeling (referred to as a transition henceforth) does not correspond to any relation in \( H \) or \( V \). The labels will not be reset after a task is completed and are carried over to the following tasks. Each state of the domain corresponds to a possible transition from an environment state to the next.

It can now be shown that the decision problem \( D_{\text{MinW}} \) has a solution of size \( |H| + |V| \) if and only if the problem \( D \) has a consistent tiling. In the first direction, if TILING has a consistent tiling, we can set the \( |H| + |V| \) words according to \( H \) and \( V \), each specifying a horizontal or vertical compatibility relation that corresponds to a set of legitimate transitions. This set of words clearly can be used to coordinate the agents’ transitions in the desired manner since every word specifies a single valid transition given the current environment state (i.e., agent locations and labels). On the other hand, if there does not exist a tiling that is consistent, there must exist two adjacent locations that could not satisfy any relation in \( H \) or \( V \) when all the locations are labeled. In such a situation, a failure will be resulted if all the plans for a given task require the agents to move along those two locations, no matter how we choose the word set. In other words, no coordination languages could have existed.

**V. Approximating \( D_{\text{MinW}} \)**

We have shown that the class of \( D_{\text{MinW}} \) problems is generally NEXP-complete. Hence, solving the problems exactly is computationally infeasible. Next, we focus on developing an approximate solution for \( D_{\text{MinW}} \). First, we formally define state abstraction used in \( D_{\text{MinW}} \).

**Definition 5** (State Abstraction). A state abstraction of a state space \( S \) is a mapping \( f : 2^S \rightarrow \{1 \ldots M\} \) where \( M \) is the number of abstract states. Where there is no ambiguity, \( M \) is also used to denote the set of abstract states.

Note that the same state may appear in multiple abstract states under this definition. When viewing the TSC in Fig. 3 as a sentence that is constructed by concatenating words of state abstractions, it may also be considered as an illustration of the relationship between abstract states and physical states. It follows that the set of words for \( D_{\text{MinW}} \) is a state abstraction. We further define a perfect state abstraction as a state abstraction that corresponds to a partition: every physical state belongs to one and only one abstract state. This means that \( f \) now becomes a surjective function for the mapping of \( S \rightarrow \{1 \ldots M\} \) such that \( f(s) \) returns the abstract state for \( s \).
Lemma 1. The set of words (i.e., $W$) of any coordination language for $D_{\text{MinW}}$ may be replaced by a perfect state abstraction, resulting in another coordination language.

Proof: This is easy to see since we can construct a perfect state abstraction from a state abstraction by constructing additional abstract states for the overlapping states, and for states not covered by the original set of words. Then, for any TSC that may be constructed by the original state abstraction, a more stringent TSC can be constructed using the perfect state abstraction, which does not affect the requirements of a coordination language (see Def. 4). This also means that this new language can still express any optimal plan for any task. Hence, the conclusion holds.

Notice, however, that the replacement may reduce the flexibility when using the language since it now contains more words and tends to be more stringent. The benefit of using perfect state abstractions is that the TSC (or sentence) constructed to express any plan is unique since any physical state is also uniquely mapped to an abstract state.

Corollary 1. The decision problem of $D_{\text{MinW}}$ with perfect state abstractions, denoted as $D_{\text{MinW}^+}$, is NEXP-complete.

The proof in Theorem 1 applies also to perfect state abstractions so this is a direct result. In spite of belonging to the same complexity class, problems with perfect state abstractions are easier to analyze, which leads to an approximate solution as we will discuss next.

For any $(I, G) \in T$ of a domain, a set of optimal plans can be constructed and denoted as $\Pi$. For any plan pair $(\pi_1, \pi_2)$ in $\Pi$ that introduces RC, we denote the first and last states where they differ in the plans as $d = (\alpha, \beta)$ ($\alpha$ and $\beta$ could be the same); $\alpha$ and $\beta$ each denotes a pair of states from $\pi_1$ and $\pi_2$, respectively. Furthermore, we use $\alpha^*$ (and similarly for $\beta^*$) to denote the three pairs of nodes that are formed pair-wisely by $\alpha$ and the previous node that is shared by the plans. See Fig. 3 for an illustration of $\alpha^*$ and $\beta^*$ with two plans. We denote the set of such $(\alpha^*, \beta^*)$’s for all $(I, G) \in T$ as $D$.

Theorem 2. A language that satisfies the following condition with perfect state abstractions is a coordination language:

$\forall (\alpha^*, \beta^*) \in D, \forall i \ f(\alpha^*[i][1]) \neq f(\alpha^*[i][2]) \text{ OR } \forall i \ f(\beta^*[i][1]) \neq f(\beta^*[i][2])$.

Essentially, the condition above requires the three states that are associated with either $\alpha^*$ or $\beta^*$ between any two plans (that introduce RC) to be pair-wisely separated in the state abstraction. Intuitively, this enables a language to always distinguish between the two plans so that the associated RC can be resolved. Next, we more formally prove this result.

Proof: First, recall that a coordination language must not express any two plans that introduce RC. In other words, all plan pairs that introduce RC must always be expressed separately by such a language. We show next that a language that satisfies the condition above always resolves RC for any given plan pair $\pi_1, \pi_2$ for any $(I, G) \in T$. Given $\pi_1, \pi_2$, we can construct a unique TSC $\zeta_1$ and $\zeta_2$, respectively, given a perfect abstraction. When the above condition is satisfied, we know that $\zeta_1$ and $\zeta_2$ must differ by at least one abstract state and hence $\pi_1, \pi_2$ will never be expressed by the same TSC (see Fig. 3 for additional explanations). As a result, the RC will always be resolved by the language. Since this holds for any plan pair for any $(I, G) \in T$, we can conclude that the language is a coordination language for the domain.

The condition described above in Theorem 2 is only a sufficient condition for a coordination language–there may exist a coordination language that does not satisfy it. As a result, a language that satisfies it may not be the exact solution for $D_{\text{MinW}^+}$. In other words, it establishes an approximate solution. The positive side is that this solution only requires solving a set of (PSPACE-complete) planning problems (of size $|T|$ to be precise), which have efficient existing solutions. For our motivating example, a perfect abstraction produced is presented in Fig. 5. Note that this example is a special case where the longest plan length is 2, implying that $\alpha$ and $\beta$ will always be the same and hence the solution returned by our approximate method is in fact the exact solution for $D_{\text{MinW}^+}$.

This is further backed up by the result in Table I. Readers may have observed that the abstraction in Fig. 5 corresponds to specifying the location of agent $B$, in which case agent $A$’s location no longer matters. Using this abstraction, the agents can specify clockwise or counterclockwise movements (of $B$) by composing the words in the desired sequences, as we initially discussed (see Fig. 1). Another note is that the perfect abstraction may not be unique. For example, $A$ and $B$ have a symmetrical relationship in our motivating example so that a similar abstraction for $A$ also exists. Finally, it is worth noting that the semantic meanings (physical states) associated with the abstraction (abstract states) in this example are drawn from hindsight; the construction is automatic based solely on the coordination requirements of the domain.

Advantages of MinW: A precondition for using a MinW as a language in our setting is that the two parties both maintain the language specifications (e.g., $W$ only with $D_{\text{MinW}^+}$), which requires knowledge alignment a priori, or learning a language [16]. The advantages of using a MinW is at least...
TABLE I: Performance of our language construction method for \( D_{\text{MinW}} \) in a multi-agent path-finding domain.

| env. size | \( |I| \) | \( |S| \) | \( M \) | time (s) | \( \alpha M \) | time (s) |
|-----------|-----|-----|-----|-------|--------|-------|
| GWC #1    | 2 × 2 | 132 | 12  | 3    | 26.2   | 8     |
| GWC #2    | 2 × 3 | 870 | 30  | 2    | ≥ 3600.0 | 13    | 0.1   |
| GWC #3    | 3 × 3 | 3112 | 72 | 3    | ≥ 3600.0 | 17   | 19.3   |
| GWC #4    | 3 × 4 | 956 | 90  | ≥ 4 | 3600.0 | 23   | 86.2   |

We focus our evaluation on a multi-agent path-finding domain that is similar to the motivating example to illustrate the complete set of features of the constructed languages. Applying our method to other domains should not be much different and similar results are expected. In our evaluation domain, the agents operate in a grid-world environment and can move to any adjacent cell or stay in the same cell in the next time step. Similarly, the agents cannot stay in the same cell or switch locations at the same time step. The optimization criterion is to minimize the makespan of the joint plan. While seemingly simple, this domain is in fact challenging for coordination since the agents can freely choose their actions according to any joint optimal plan as long as they end up in their respective goal locations. This means that an agent may choose to wander around in any preferred way or stay at the goal or any intermediate locations while waiting for the other agent to reach its goal. Hence, the domain in fact represents a challenging setting for language construction. The results were obtained on a MacBook Pro with Intel Core i7 CPU 2.80GHz with 16GB of RAM using a Java implementation.

A. Language Construction

For the first set of results, we evaluate the performance of our language construction method for \( D_{\text{MinW}} \), using the approximate solution described in Theorem 2, and compare it with a brute-force method that computes the exact solution for \( D_{\text{MinW}} \). The brute-force method checks through all possible abstractions and returns the one with the smallest (word) size that is a coordination language. It always starts with the smallest size abstraction and gradually increases the size. To further reduce computation, in addition to following Theorem 2, we considered only \( \alpha^* \) and implemented a greedy solution for determining the smallest abstraction that satisfies the condition in Theorem 2, i.e., adding any remaining states to an abstract state whenever possible until nothing more can be added. Table I presents the results, where the top part corresponds to standard grid-worlds and the bottom part corresponds to grid-worlds with a non-traversable center (GWC)–an agent may only go around the center.

To verify that using TSC increases the flexibility of the listener, using the same setting as above, we compute the number of plans that are available to the listener after receiving a TSC from the speaker. The speaker always chooses the first plan that is found and then converts it to a TSC according to the language. The result is presented in Fig. 6. The average communication saving is 33.3% among the 784 plans or 27.3% among all the 956 plans, which is rather impressive given this simple setting.

B. Communication Savings

To illustrate the benefit of communication savings for the speaker, we compare our method with a baseline approach that communicates the entire plan. The cost of communication is determined by the length of a plan, and in our method the length of a TSC. We chose the setting for GWC #3 in Table II since the plans are longer and hence the effect would be clearer. The result is presented in Fig. 6. The average communication saving is 33.3% among the 784 plans or 27.3% among all the 956 plans, which is rather impressive given this simple setting.

C. Increased Flexibility

To verify that using TSC increases the flexibility of the listener, using the same setting as above, we compute the number of plans that are available to the listener after receiving a TSC from the speaker. The speaker always chooses the first plan that is found and then converts it to a TSC according to the language. The result is presented in Fig. 6. The increase in flexibility is notable: the average number of plans available is 14.7 among the 774 plans or 12.1 among all the 956 plans. (compared to 1 with the baseline approach)!
E. Coordination Game

In this experiment, we test the benefits of the flexibility provided to the listener in a game setting. In this game, given a random task, the speaker first chooses its optimal plan randomly. At the same time, the listener randomly chooses $k$ plans as its preferences out of all the possible plans for the task. The speaker then acts as a proposer that either communicates its plan or the TSC to the listener. While respecting the speaker’s preferences, if there exists a plan that the listener can choose from that is also in its own preferences, the listener accepts the proposal and the game ends. Otherwise, the game continues from the beginning (the speaker re-picks its plan and listener re-picks its preferences) until the proposal of the speaker is eventually accepted. We are interested to see how many iterations (i.e., negotiations) the agents play this game under the settings where plans and TSCs are communicated by the speaker, respectively. We run this game on GWC #1 described in Table I. The results are averaged over 100 runs. We can see that both settings the number of iterations decreases as $k$ increases, which is expected. However, these numbers remain relatively consistent and much smaller for the TSC case which clearly shows the benefit of the flexibility enabled by TSCs.

VII. CONCLUSIONS AND DISCUSSIONS

In this paper, we provided a preliminary and formal treatment of the language construction problem for autonomous agents from a new computational perspective. The focus of this initial investigation was on a task planning setting where one agent directly communicates an abstract plan to the other agent to achieve coordination. We viewed language construction as a fundamental requirement for resolving miscoordination. Such a view enables us to compute a language from the ground up by mapping symbols to physical states. Furthermore, we introduced the notion of coordination language and discussed its theoretical properties. Our later discussion was focused on one form of coordination languages that minimize the word set with a given set of operators, or MinW. It was shown that even such a restricted form is NEXP-complete. An approximate solution was then developed to convert this NEXP-complete problem into a set of planning problems. Finally, evaluations of our method were provided.

This preliminary discussion herein opens up plentiful opportunities for future investigation. For example, discussions for removing some of the assumptions, such as plan optimality, goal state specification, cooperative setting, plan commitment, and the requirements of a coordination language in Def. 4 would significantly improve the applicability of the constructed languages. For example, instead of requiring no RC to be introduced in Def. 4, we may require only that a
significant portion of the plans do not introduce RC by defining a robustness measure similar to that used in [17, 29]. Another interesting direction is to address the evolution of languages in new environments with additional (and potentially heterogeneous) agents. Although we can naively extend our approach to handle such cases, the computation would scale exponentially. Along this line, methods may be explored to enable online and incremental learning. Better approximate solutions are also desirable. Finally, it is worth noting that although it is not the intention of this work, the proposed computational view on language construction from coordination requirements could shed new light on the origin of languages [33].

REFERENCES

[1] David Abel, D Ellis Hershkowitz, and Michael L Littman. Near optimal behavior via approximate state abstraction. arXiv preprint arXiv:1701.04113, 2017.

[2] Nicholas Allott. Game Theory and Communication, pages 123–152. Palgrave Macmillan UK, London, 2006. ISBN 978-0-230-28589-7. doi: 10.1057/9780230328589.4. URL https://doi.org/10.1057/9780230328589.4

[3] Jacob Andreas, Anca Dragan, and Dan Klein. Translating neuralese. arXiv preprint arXiv:1704.06960, 2017.

[4] Mihai Barbuceanu and Mark S Fox. Cool: A language for describing coordination in multi agent systems. In ICMAS, pages 17–24, 1995.

[5] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. Mathematics of operations research, 27(4):819–840, 2002.

[6] Angelo Cangelosi and Domenico Parisi. Simulating the evolution of language. Springer Science & Business Media, 2012.

[7] Noam Chomsky. Three factors in language design. Linguistic inquiry, 36(1):1–22, 2005.

[8] Noam Chomsky. The minimalist program. MIT press, 2014.

[9] Morten H Christiansen and Simon Kirby. Language evolution. OUP Oxford, 2003.

[10] William Cushing, Subbarao Kambhampati, and Daniel S Weld. When is temporal planning really temporal? In Proceedings of the 20th international joint conference on Artificial intelligence, pages 1825–1859, 2007.

[11] E Allen Emerson. Temporal and modal logic. In Formal Models and Semantics, pages 995–1072. Elsevier, 1990.

[12] Kutluhan Erol, James Hendler, and Dana S Nau. HTN planning: Complexity and expressivity. In AAAI, pages 1123–1128, 1994.

[13] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3):189 – 208, 1971. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(71)90010-5. URL http://www.sciencedirect.com/science/article/pii/0004370271900105

[14] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. Kqml as an agent communication language. In CIKM, pages 456–463. ACM, 1994.

[15] Jon Gauthier and Igor Mordatch. A paradigm for situated and goal-driven language learning. CoRR, abs/1610.03585, 2016. URL http://arxiv.org/abs/1610.03585

[16] Dedre Gentner and Laura L Namy. Analogical processes in language learning. Current Directions in Psychological Science, 15(6):297–301, 2006.

[17] Matthew L. Ginsberg. Approximate planning. Artificial Intelligence, 76(1):89 – 123, 1995. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(94)00077-E. URL http://www.sciencedirect.com/science/article/pii/000437029400077E

[18] Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In AAMAS, pages 137–144, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8. doi: 10.1145/860575.860598. URL http://doi.acm.org/10.1145/860575.860598

[19] Ze Gong and Yu Zhang. Temporal spatial inverse semantics for robots communicating with humans. In ICRA. IEEE, 2018.

[20] Stevan Harnad. The symbol grounding problem. Physica D: Nonlinear Phenomena, 42(1-3):335–346, 1990.

[21] Marc D Hauser, Noam Chomsky, and W Tecumseh Fitch. The faculty of language: What is it, who has it, and how did it evolve? science, 298(5598):1569–1579, 2002.

[22] Subbarao Kambhampati, Craig A Knoblock, and Qiang Yang. Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. Artificial Intelligence, 76(1-2):167–238, 1995.

[23] Simon Kirby, Tom Griffiths, and Kenny Smith. Iterated learning and the evolution of language. Current opinion in neurobiology, 28:108–114, 2014.

[24] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. IJRR, 31(3):360–375, 2012.

[25] George Konidaris, Leslie Kaebbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. In AAAI, 2014.

[26] Jonas Kvarnström and Patrick Doherty. Talplanner: A temporal logic based forward chaining planner. Annals of mathematics and Artificial Intelligence, 30(1-4):119–169, 2000.

[27] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In Experimental Robotics, pages 403–415. Springer, 2013.

[28] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In AAAI, 2018.

[29] Tuan A Nguyen, Subbarao Kambhampati, and Minh Do. Synthesizing robust plans under incomplete domain models. In NIPS, pages 2472–2480, 2013.

[30] Rohan Paul, Jacob Arkin, Nicholas Roy, and Thomas
[31] Stefan Poslad. Specifying protocols for multi-agent systems interaction. *TAAS*, 2(4):15, 2007.

[32] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[33] Mark Steedman. Plans, affordances, and combinatory grammar. *Linguistics and Philosophy*, 25(5-6):723–753, 2002.

[34] Luc Steels. Evolving grounded communication for robots. *Trends in cognitive sciences*, 7(7):308–312, 2003.

[35] Luc Steels. Modeling the cultural evolution of language. *Physics of life reviews*, 8(4):339–356, 2011.

[36] Luc Steels. Grounding language through evolutionary language games. In *Language Grounding in Robots*, pages 1–22. Springer, 2012.

[37] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *NIPS*, pages 2244–2252, 2016.

[38] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *AIJ*, 112(1):181–211, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00052-1. URL http://www.sciencedirect.com/science/article/pii/S0004370299000521

[39] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashish Gopal Banerjee, Seth Teller, and Nicholas Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI magazine*, 32(4):64–76, 2011.

[40] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashish Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011.

[41] Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. Asking for help using inverse semantics. In *RSS*, Berkeley, USA, July 2014. doi: 10.15607/RSS.2014.X.024.

[42] Moshe Y Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for concurrency*, pages 238–266. Springer, 1996.

[43] Li Wang and Qiao Guo. Coordination of multiple autonomous agents using naturally generated languages in task planning. *Applied Sciences*, 9(17):3571, 2019.

[44] Ping Xuan, Victor Lesser, and Shlomo Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *AAMAS*, pages 616–623. ACM, 2001.

[45] Yu Zhang, Kangjin Kim, and Georgios Fainekos. Discof: Cooperative pathfinding in distributed systems with limited sensing and communication range. In *Distributed Autonomous Robotic Systems*, pages 325–340. Springer, 2016.

[46] Yu Zhang, Sarath Sreedharan, and Subbarao Kambhampati. A formal analysis of required cooperation in multi-agent planning. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.