# ACoL: From Abstractions to Grounded Languages for Robust Coordination of Task Planning Robots

Yu Zhang
School of Computing and Augmented Intelligence (SCAI)
Arizona State University
Tempe, Arizona 85281
Email: yzhan442@asu.edu

***Abstract*— In this paper, we consider a first step to bridge a gap in coordinating task planning robots. Specifically, we study the automatic construction of languages that are *maximally flexible* while being *sufficiently explicative* for coordination. To this end, we view language as a machinery for specifying temporal-state constraints of plans. Such a view enables us to reverse-engineer a language from the ground up by mapping these *composable* constraints to words. Our language expresses a plan for any given task as a "*plan sketch*" to convey just-enough details while maximizing the flexibility to realize it, leading to robust coordination with optimality guarantees among other benefits. We formulate and analyze the problem, provide approximate solutions, and validate our approach under various scenarios to shed light on its applications.**

## I. Introduction

To facilitate explicit coordination via communication between robots in distributed systems, a key consideration is the adoption of a "*language*" that the robots can all speak. Such a language often relies on words with predefined meanings that are designed by human users [1], [2], [3], [4]. However, such languages tend to be either too rigid or too forgiving, leading to brittle solutions, excess negotiation costs, or unexpected coordination issues (e.g., deadlocks). In this paper, as a first step, we consider to bridge the gap for task planning robots using symbolic planning. Traditional methods for explicit coordination via communication in distributed systems with planning agents can be divided into two classes:

1) *Centralized plan and distributed execution*: provides optimality guarantees except when approximate solutions are considered [5], [6], [7]. Note that the planning process may be centralized or distributed [6]. Explicit coordination in this class involves broadcasting the centralized plan in the planning language and sometimes exchanging messages as stipulated by the plan during plan execution. This approach results in brittle solutions (i.e., a single agent changing its part of plan requires the entire plan to be updated) among other limitations.

2) *Distributed plans and distributed execution*: provides no guarantee of optimality in general [8], [9]. Note that distributed plans imply that the planning process is distributed. Methods in this class are often rule or local-search based [10], [11], making them adaptive to local changes and easy to implement. For example, [12] addressed similar coordination problems via replanning and social rules (such as *waiting*). For explicit coordination, a language is often designed manually on a case-by-case basis, which is prone to unexpected coordination issues (e.g., deadlocks).

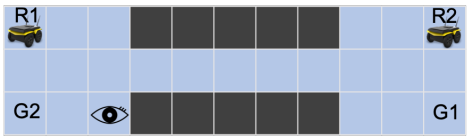

Fig. 1: Motivating scenario involving two navigational robots.

Our work serves as a middle ground that bridges the two classes and combines their advantages, contributing a novel perspective for explicit coordination between task planning agents. Consider the navigation scenario in Fig. 1 that involves two robots, $R_1$ and $R_2$, in a gridworld. Each cell can only accommodate a single robot at a time and the darker cells are obstacles. The robots are tasked to reach their goal locations ($G_1$ and $G_2$), respectively, in the shortest timespan while avoiding collisions. They have a limited sensing range and communication is costly. During plan execution, there may be locations of interest popping up at random places that require either one of the robots to visit (denoted by the eye sign). Even without considering the locations of interest, the problem is difficult for the second class of methods: the robots must coordinate before one of them enters the narrow pathway so methods based on local information only would not work well (i.e., leading to deadlocks). With changing task configuration and unpredictable locations of interest, neither would be assigning fixed priorities to the robots (e.g., always letting $R_1$ go through first). While these issues are not present in the first class where the robots must coordinate a plan before execution, whenever some location of interest pops up, the robots must re-coordinate a plan, significantly increasing the communication and coordination cost.

While similar to the first class, robots in our approach coordinate by communicating "*plan sketches*" that guarantee optimality while maximizing flexibility to reduce the need for replanning and re-coordination (thus differing from work on replanning or plan repair for making replanning more efficient [13]). In the scenario above, the robots can communicate that "$R_2$ *to wait for* $R_1$" without specifying the exact plan to be followed, so that robot $R_2$ later (instead of $R_1$ even though $R_1$ would detect the location of interest first, since that would require $R_1$ to wait for $R_2$) can adjust locally to visit the location of interest even if its original plan does not pass through it. A crucial property is to enable each

robot to make local changes without any negative impact to the (global) makespan as long as the updated plan is still consistent with the plan sketch.

To this end, we view language as a machinery for specifying plan constraints. A language thus specifies a *plan space abstraction* where a sentence in the language (i.e., a plan sketch) specifies a set of plans. The robots all commit to the same set of plans as a result of coordination (i.e., one robot communicates a plan sketch to the others). To guarantee the feasibility of this approach, given that the robots may be unaware of the local changes made by the others, one of key challenges is for the plans in this set to not introduce *miscoordinations*; to maximize flexibility, the number of plans should be maximized. Since different sets will be specified for different tasks, we instead minimize the number of words in the language, resulting in a *minimal language*, referred to as a minimal **Agent Coordination Language** (ACoL). In our approach, we associate words in the language with temporal-state constraints that are composable. We show that searching for a minimal language is NEXP-complete. In light of this result, we develop two approximate solutions: the first solution guarantees the prevention of miscoordinations while the second solution is tractable and provides more flexibility at the cost of such a guarantee. Finally, we validate the benefits of the languages under various application scenarios. Hence, our contribution is both theoretical and practical in nature (with a 2-page version appeared earlier [14]).

## II. Related Work

The emergence and evolution of language has been studied extensively in evolutionary and computational linguistics [15], [16], [17]. Most prior work there has considered the problem in the context of evolutionary games [18], and often in an iterated learning setting [19]. Steedman's work [20] 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. Our work introduces a framework that establishes the connection between a *coordination language* and the underlying planning language where the new language arises solely from the need for coordination. From a computational perspective, our work is also inspired by a prior study that considers the incentive of communication in a game-theoretical framework [21].

A language often represents a structured symbolic system mapping symbols to semantic meanings that can be grounded in the environment [22], [18], [23], [24], [25]. In this work, we instead reverse engineer the process by considering the mapping from temporal-state constraints to symbols for language construction. These symbols (i.e., words) are then used to form sentences, which introduce a *plan space abstraction* to resolve miscoordinations. The idea of applying abstraction to planning problems is not new. Most prior work has focused on state abstraction for problem decomposition, which has been well studied in both path and task planning methods [26], [27], [28]. Such decomposition has also been shown to benefit communication and coordination in multi-agent planning [29], [30]. The temporal-state constraints for plan space abstraction used in our approach resemble options in semi-MDP and LTL expressions in temporal logic [31], [32], [33]. However, these prior approaches have mostly focused on applying plan space abstraction to planning [34], [31], or learning such abstraction for problem decomposition [35], [36]. We instead consider plan space abstraction for coordination. Coordination program synthesis with LTL and CSP [37] is particularly relevant but focused on only a specific problem instance. It thus differs substantially from our work targeting a general machinery for robust coordination under any instances, even though the coordination mechanism under a given instance may appear similar.

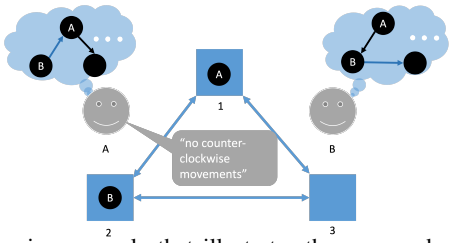

Fig. 2: Running example that illustrates the proposed plan sketch and the coordination process involved.

Our problem may also appear similar to the problem of learning or planning to communicate. There exists work on developing communication schemes to maximize team performance, where the problem could be considered either in a planning or learning setting [38], [39], [40], [41]. For example, a communication scheme can arise from a Dec-POMDP framework where communicating actions are modeled specifically [39]. Such actions essentially augment the observation function. The focus there is on exploiting the additional information that can be associated with communication to maximize the expected return. No explicit connection is established between the communication symbols and their semantic meanings (i.e., the language is not grounded).

## III. Problem Formulation

We adopt a **joint representation while assuming discrete state and action spaces**: whenever referring to a plan, we refer to a *joint and optimal* plan. The sequence of local actions from each robot's perspective is referred to as a *subplan*. A plan is constituted of robot subplans, which, in turn, can form into plans (details later). Similarly, states and actions refer to joint states and actions. Substate and subaction refer to each robot's local state and action. Proofs are included in the full version [42].

### A. Problem Setting

We focus our discussion on the problem setting that involves only two robots with the following assumptions:

- $A1$: Both robots have access to the full planning domain model, the task information, and sufficient computational resources, such that either robot can compute plans independently.
- $A2$: Each robot may not observe the other robot or changes to the environment made by the other during plan execution (so as to focus on explicit coordination).

$A1$ results in a special case of cooperative multi-agent planning setting where agents maintain accurate beliefs about the domain models of the others [43]. See Sec. V about relaxing these assumptions. Without coordination, each robot may choose any candidate plan for the task and execute the corresponding subplan. Coordination is needed to ensure that the robots do not choose plans that lead to *miscoordinations*, i.e., situations when choosing different plans leads to suboptimality or task failures. To allow local changes (i.e., changes to subplans) that are critical for flexibility, part of the aim is to retain as many candidate plans as possible. However, more candidate plans will more likely lead to miscoordinations, creating a challenging tradeoff. Intuitively, the solution is to have the robots both commit to the same set of plans (a subset of all candidate plans for the task) via coordination. The challenge that remains is to identify the largest sets of plans for different tasks where local changes can be *independently* made without affecting the (global) task performance.

### B. Running Example

We introduce a running example in Fig. 2 with two robots $\{A, B\}$ and three locations $\{1,\ 2,\ 3\}$. The robots cannot switch locations or stay in the same location in the same time step to avoid collisions. Without coordination, each robot may choose any candidate plan for the task and follow its corresponding subplan. In a situation where the plans are chosen differently as shown in the thought bubbles, it would lead to a miscoordination (i.e., a collision). We consider a coordination process where either robot communicates a "*plan sketch*" (specifying a set of candidate plans) for both robots to commit to. A plan sketch may correspond to a set of plans having certain properties. For example, $A$ can tell $B$ to *not* move counter-clockwise, which resolves the miscoordination illustrated in Fig. 2. In contrast to $A$ directly communicating its plan to $B$, this way, $B$ has the flexibility to either wait first and then move, or move first and then wait, resulting in more robust coordination.

### C. Preliminaries

We model the planning domain based on a slightly modified STRIPS model [44] as $M = (P, A)$, where $P$ is the set of propositional state variables and $A$ is the set of actions. Each action $a \in A$ specifies a pair of robot subactions such that $a = \langle a^A, a^B \rangle$, where $\langle\rangle$ denotes an ordered set. Each action $a$ 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 consider a set of candidate tasks in the domain for our language formation problem, which represents all relevant tasks to the robots. Each candidate task is in the form of a pair $(I, G)$, where $I \in 2^P$ and $G$ specifies the propositions required to be present in the goal state. Each $(I, G)$ pair introduces a planning problem $O = (P, A, I, G)$. A plan $\pi$ is a sequence of actions:

$$\pi = \langle \langle a_1^A, a_1^B \rangle .. \langle a_n^A, a_n^B \rangle \rangle \tag{1}$$

where $n$ is the length of the plan. $\pi$ may also be specified as the combination of two subplans as $\pi = \langle \pi^A, \pi^B \rangle$, where $\pi^A = \langle a_1^A, .., a_n^A \rangle$ denotes the subplan for robot $A$ and $\langle\rangle$ here may also be viewed as an operator that element-wisely combines $\pi^A$ and $\pi^B$ into a plan. When combined, $\pi^A$ and $\pi^B$ are assumed to be aligned from the first step and onward, and the shorter subplan is padded with *idle* subactions.

Given a domain model $M$, the resulting state after executing plan $\pi$ in state $s$ is determined by the transition function $\gamma$, defined as follows where '$\cdot$' denotes concatenation:

$$\gamma(\pi, s) = \begin{cases} s & \text{if } \pi = \langle\rangle \\ \gamma(\langle a \rangle, \gamma(\pi', s)) & \text{else } \pi = \pi' \cdot \langle a \rangle \end{cases} \tag{2}$$

The transition function $\gamma$ for an action sequence with a single action $a$ and state $s$ is defined as:

$$\gamma(\langle a \rangle, s) = \begin{cases} (s \setminus Del(a)) \cup Add(a) & \text{if } Pre(a) \subseteq s \\ s & \text{otherwise} \end{cases} \tag{3}$$

Given a planning problem $O = (P, A, I, G)$, an action sequence $\pi$ is a plan for $O$ iff $\gamma(\pi, I)$ entails $G$ *and* $\pi$ has the minimal cost. For simplicity, we assume that every action has a cost of 1 and hence a plan for $O$ minimizes the makespan to satisfy $G$ from $I$. A plan $\pi$ also introduces a state sequence, denoted by $\pi_S$, such that the first element $\pi_S[0] = I$ and the last $\pi_S[n] = \gamma(\pi, I)$ entails $G$ with a plan length $|\pi| = n$.

### D. Required Coordination

We next formally define *required coordination* that introduces miscoordinations. Under our problem setting, a miscoordination may occur when robots choose different candidate plans. Given a planning problem $O = (P, A, I, G)$, $\Pi(O)$ denotes the set of all candidate plans for $O$. Without coordination, either robot can choose any plan in $\Pi(O)$.

*Definition 1 (Required Coordination (RC)):* Given a planning problem $O = (P, A, I, G)$, a required coordination is the following condition: $\exists \pi_1, \pi_2 \in \Pi(O)$ $(\pi_1 \neq \pi_2)$, $\langle \pi_1^A, \pi_2^B \rangle \notin \Pi(O) \mid \langle \pi_2^A, \pi_1^B \rangle \notin \Pi(O)$, where $\pi_1 = \langle \pi_1^A, \pi_1^B \rangle, \pi_2 = \langle \pi_2^A, \pi_2^B \rangle$.

Intuitively, an RC condition defines a situation where there are two different plans $\pi_1, \pi_2 \in \Pi(O)$ for problem $O$ but the recombination of the subplans does not belong to $\Pi(O)$ (i.e., not a plan for $O$). If the set of candidate plans that the robots commit to includes these two plans, a miscoordination may occur when the robots choose $\pi_1$ and $\pi_2$, respectively. In such cases, we say that the two plans *introduce* RC. Two different plans do not necessarily introduce RC (Fig. 3).

*Proposition 1:* Given a domain model $M = (P, A)$ and a set of candidate tasks $\mathcal{T}$, coordination is necessary if and only if the following holds: $\exists \pi_1, \pi_2 \in \Pi(O)$, $(\pi_1, \pi_2)$ introduces RC, where $(I, G) \in \mathcal{T}$ and $O = (P, A, I, G)$.

Given any planning problem $O$ above, if no plan pair exists that introduces RC, we consider the following two possible cases. 1) No plan or only a single plan exists: in such a case, either the robots will both fail to find a plan or they will find the same plan. No coordination is necessary. 2) There are multiple plans but none of them pair-wisely introduce RC. In such a case, either robot can select any candidate plan for $O$, and the recombined plans are guaranteed to be in $\Pi(O)$ (Def. 1) and it is hence miscoordination-free.

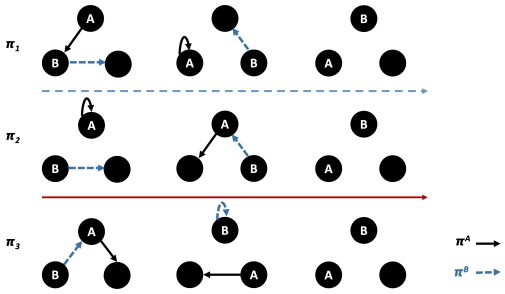


Fig. 3: Three different plans for the task in Fig. 2: the first two do not introduce RC: subplan $\pi_1^A$ (black arrows) in $\pi_1$ can be switched with its counterpart ($\pi_2^A$) in $\pi_2$ and recombined with subplan $\pi_2^B$ (blue dashed arrows) without introducing miscoordinations. Both $\pi_1$ and $\pi_2$, however, introduce RC with $\pi_3$.

In Fig. 3, $\pi_1$ and $\pi_2$ do not introduce RC. Hence, had these two plans being the only plans in the set of candidate plans that the robots commit to (as a result of coordination) for a task, no additional coordination would be necessary. This observation already hints on our language construction: a language will be required to *always* express $\{\pi_1, \pi_2\}$ and $\{\pi_3\}$ differently under the task in Fig. 2.

### E. Agent Coordination Language (ACoL)

The ability to separately express different sets of candidate plans requires a language to specify constraints on plans, essentially forming plan state abstraction. Since plans are temporal state sequences, we consider words as symbols that map to temporal-state constraints. These symbols can be further combined to form sentences for expressing more complex constraints. We use : to indicate a range of indices (inclusive at both ends). For example, $\pi_S[x:y]$ returns the set of states in $\pi_S$ indexed from $x$ to $y$. An *abstract state* $S_i$ corresponds to a subset of the state space $S$ (i.e., $S_i \subseteq S$).

*Definition 2 (Temporal-State Constraint (TSC)):* A TSC over a state space $S$ specifies a constraint $\zeta$ in the form of $\zeta = S_1 \dots S_i \dots S_m$, where $S_i \subseteq S$ is an abstract state.

A plan $\pi$ under the same $S$ *satisfies* a TSC $\zeta$ if there exists a set of strictly monotonically increasing integers $j_0, j_1, j_2 \dots j_m$ with $j_0 = -1$ and $j_m = |\pi|$, such that $\pi_S[j_{i-1}+1 : j_i] \subseteq S_i$ and $\pi_S[j_{i-1}] \notin S_i$ and $\pi_S[j_i+1] \notin S_i$ $(n \geq m \geq i \geq 1)$. Intuitively, a TSC breaks a state sequence into segments such that the states in each segment belong to the same abstract state while their immediately adjacent states (if present) belong to other abstract states. Notice that a plan may satisfy multiple TSCs if the abstract states are allowed to overlap; otherwise, the TSC for a plan will be unique. An illustration of a TSC is provided in Fig. 4. Alternatively, we also refer to that a TSC $\zeta$ *expresses* a plan $\pi$ (denoted by $\zeta \sqsupseteq \pi$), if $\pi$ satisfies $\zeta$. One may use more expressive constraints for language formation, such as specified by LTL and CTL [32], [33]. We choose TSCs since they are easier to analyze while still being extensible.

*Definition 3 (Language):* A language for a domain $M = (P, A)$ is a pair $\mathcal{L} = (\mathcal{W}, \mathcal{O})$, where $\mathcal{W}$ is the vocabulary and $\mathcal{O}$ is the set of compositional operators that can be applied to connect the words. Each word $w \in \mathcal{W}$ is a TSC over $S = 2^P$ and TSCs must remain closed under $\mathcal{O}$.

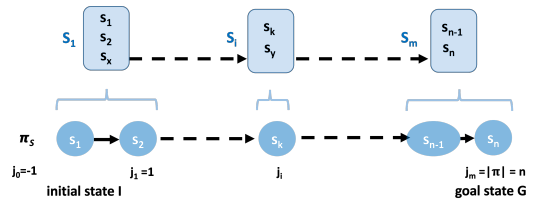


Fig. 4: An illustration of a TSC. The top shows the abstract states associated with the TSC and the bottom a plan as a state sequence that satisfies it. The rectangle nodes represent abstract states with their included ground states shown inside.

Operators are applied to connect the words in a language to form more complex TSCs, which are also referred to as *sentences*. A given language thus introduces a rich set of TSCs for the domain. Not all languages are of interest to the robots. To be useful, a language must be able to relax RCs for any candidate tasks (following Proposition 1). Any language with such a capability is referred to as a *coordination language*. In the following, with a slight abuse of notation, we denote any TSC $\zeta$ that can be expressed by a language $\mathcal{L}$ as $\zeta \in \mathcal{L}$.

*Definition 4 (Coordination Language):* A language $\mathcal{L}$ for a domain $M = (P, A)$ is an (agent) coordination language under a set of candidate tasks $\mathcal{T}$ if the following condition holds: $\forall \pi \in \Pi(O)$ where $O = (P, A, I, G)$ and $(I, G) \in \mathcal{T}$, 1) $\exists \zeta \in \mathcal{L}, \zeta \sqsupseteq \pi$, and 2) the subset of plans in $\Pi(O)$ that are expressed by $\zeta$ must be RC-free, or more formally, $\{\pi | \zeta \sqsupseteq \pi \wedge \pi \in \Pi(O)\}$ pair-wisely introduce no RC.

More intuitively, since a sentence may express multiple plans, a coordination language ensures that, under any candidate task, 1) any plan is expressible by a sentence in the language (*completeness*), and 2) the plans that are expressed by any such sentence do not pair-wisely introduce RC (*RC-free*). Hence, the language can be used to coordinate the robots under any candidate task. Finding a coordination language is not difficult. For example, when we use words for grounded actions and concatenate them, we can express any plan exactly. Such a language, however, is undesirable since it is too rigid for coordination. For more flexibility, we would like to maximize the number of plans expressed. Since the situation may differ from task to task, instead, we consider the problem to minimize the size of the vocabulary, which also aligns with language design principles [45], [46], [47]. To simplify its formal analysis, we only consider concatenation $(\cdot)$ and assume words formed by abstract states only (i.e., $\mathcal{W} \subseteq 2^S$), which introduce a *state abstraction*. Note that TSCs are naturally closed under $\cdot$.

*Definition 5 (State Abstraction):* A state abstraction is a set of abstract states, $Z = \{S_z\}$, over a subset $S'$ of the state space $S$, corresponding to a many-to-many mapping $f : S' \to Z$, where a state mapping to an abstract state belongs to that abstract state.

*Theorem 1:* The problem of deciding whether a coordination language for domain $M$ under a set of candidate task $\mathcal{T}$ exists with vocabulary $\mathcal{W}$ as a state abstraction of a minimal size $K$ and $\mathcal{O} = \{\cdot\}$ (concatenation), or denoted by $D_{MinW} = (M, \mathcal{T}, K)$, is NEXP-complete.

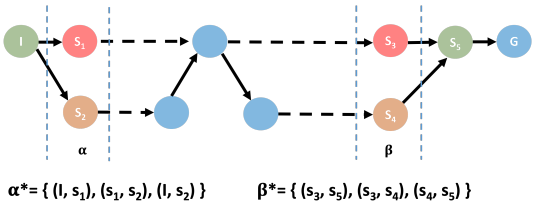


Fig. 5: Determining $(\alpha^*, \beta^*)$ with two plans (top and bottom) that introduce RC for a given task. When $\{I,\ s_1,\ s_2\}$ (or $\{s_3,\ s_4,\ s_5\}$) are pair-wisely separated into different abstract states, the two sentences for expressing these two plans will be different since they must represent $s_1$ and $s_2$ (or $s_3$ and $s_4$) as different abstract states with the preceding (or trailing) abstract state shared between the sentences, thus relaxing the RC.

### F. Finding Coordination Languages

Next, we focus on developing an approximate solution for $D_{MinW}$. We define a *perfect state abstraction* as a state abstraction that introduces a partition of the state space: every state belongs to one and only one abstract state. $f$ in Def. 5 becomes a surjective mapping over the entire state space $S$.

*Lemma 1:* Given a coordination language $(\mathcal{W}, \{\cdot\})$ for domain $M$ under a set of candidate tasks $\mathcal{T}$, there is another coordination language, $(\mathcal{W}', \{\cdot\})$, where $\mathcal{W}'$ is perfect.

*Theorem 2:* The decision problem of $D_{MinW}$ with a perfect vocabulary, denoted by $D_{MinW^+}$, is NEXP-complete.

To simplify notations, we also use $D_{MinW^+}$ to denote the problem of searching for a coordination language with a minimal perfect state abstraction when there is no ambiguity. Note that a language that uses a perfect state abstraction may have a reduced flexibility since it tends to create more stringent TSCs. Using a perfect state abstraction implies that the TSC for expressing any plan is *unique*.

For each candidate task $(I, G) \in \mathcal{T}$ of a domain $M = (P, V)$, we can compute $\Pi(O)$ for the induced planning problem $O = (P, V, I, G)$. For any plan pair $(\pi_1, \pi_2)$ in $\Pi(O)$ that introduces RC, we denote the first and last places where they differ as a pair $d = (\alpha, \beta)$. $\alpha$ and $\beta$ can be the same; $\beta$ does not always exist since the ground goal states for $\pi_1$ and $\pi_2$ may be different. $\alpha$ ($\beta$) includes a state from $\pi_1$ and a state from $\pi_2$. We use $\alpha^*$ ($\beta^*$) to denote the three pairs of states that are formed pair-wisely by $\alpha$ ($\beta$) with the previous (next) state shared by the plans. Fig. 5 illustrates $\alpha^*$ and $\beta^*$ with two plans. We collect $(\alpha^*, \beta^*)$ for all $(\pi_1, \pi_2)$ and for all $(I, G) \in \mathcal{T}$ as $\mathcal{D}$.

*Theorem 3:* Given the $\mathcal{D}$ above induced from a given domain $M$ and a set of candidate tasks $\mathcal{T}$, if a perfect state abstraction $\mathcal{W}$ (with its mapping denoted by $f^+$) satisfies the condition that $\forall(\alpha^*, \beta^*) \in \mathcal{D}, \forall i\ f^+(\alpha^*[i][0]) \neq f^+(\alpha^*[i][1])$ or $\forall i\ f^+(\beta^*[i][0]) \neq f^+(\beta^*[i][1])(i \in \{0, 1, 2\})$, the language of $(\mathcal{W}, \{\cdot\})$ is a coordination language for $M$ under $\mathcal{T}$.

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 the language to always distinguish between the two plans so that the associated RC can be relaxed. Using Theorem 3, we can convert the problem of searching for a minimal coordination language to the problem of finding a minimal perfect state abstraction that satisfies the condition therein. See also Alg. 1.

**Algorithm 1:** Approx. minimal language for $D_{MinW^+}$

**input :** $M, \mathcal{T}, \mathcal{D} = \emptyset$
**forall** *(I, G)* $\in \mathcal{T}$ **do**
  Formulate $O = (M, I, G)$
  Compute $\Pi(O)$
  **forall** $\pi_1, \pi_2 \in \Pi(O)$ **do**
    **if** *RC is introduced* **then**
      Determine $\alpha^*$ and $\beta^*$
      Add $(\alpha^*, \beta^*)$ into $\mathcal{D}$
Find an $f^+$ that satisfies the condition in Theorem 3
Extract the perfect state abstraction $\mathcal{W}$ from $f^+$
**return** $\{(\mathcal{W}, \{.\})\}$

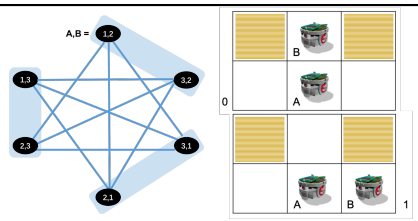


Fig. 6: **Left**: A possible set of state pairs (connected by edges) required to be separated by a perfect state abstraction in a coordination language for our running example, as determined by the condition in Theorem 3. Each node is a state with the first and second numbers representing the locations of $A$ and $B$, respectively. The blue clusters present a possible vocabulary for the language. **Right**: Brown cells are non-traversable spaces. Two of the three abstract states (labeled by '0' and '1' above) for this domain contain a single ground state. The remaining ground states all belong to the third abstract state ('2').

The condition described above in Theorem 3, however, is only a sufficient condition for a coordination language–there may exist a coordination language that does not satisfy it. The positive side is that this solution only requires solving a set of planning problems (of size $|\mathcal{T}|$ to be precise), which have efficient existing solutions. Existing planning methods that return all plans, such as m-A* [48] or DFBB [49], can be used. For our running example in Fig. 2, a perfect state abstraction produced by Alg. 1 is presented in Fig. 6 (left). Readers may have observed that the words in Fig. 6 (left) corresponds to specifying the location of robot $B$. The robots can use these words to compose sentences to specify clockwise or counterclockwise movements for $B$, consistent with our earlier discussion of Fig. 2.

### G. Tractable Approximation

The approximate solution in Alg. 1 guarantees completeness and RC-freeness when using the computed language for coordination. However, it is largely intractable due to the requirement of computing all optimal plans for each candidate task $(I, G) \in \mathcal{T}$. A key challenge for computing an optimal plan is that it can be exponentially long. As a result, we propose another approximate solution that only considers plans up to a certain length limit $k$. Computing all optimal plans for all $(I, G_k) \in \mathcal{T}_k$ ($(I_k, G) \in \mathcal{T}_k$) where

$G_k(G)$ is reachable from $I(I_k)$ in $k$ steps or under for all $(I, G) \in \mathcal{T}$ becomes polynomial. By replacing $\mathcal{T}$ with $\mathcal{T}_k$ in Alg. 1 (essentially considering only the first and last $k$ steps of any optimal plan) and combining with a greedy heuristic (Sec. IV) for finding $f^+$, we have a polynomial solution. The resulting language is still complete but no longer guarantees RC-freeness. However, we will show in our evaluations that the resulting language may be more desirable in practice.

**Practical Note:** Using a coordination language requires both robots to first "learn" the language, similar to how people learn the same language to communicate [50]. Learning a coordination language involves the maintenance of the language specifications (i.e., $(\mathcal{W}, \mathcal{O})$), such as the mapping from ground states to words (e.g., $f^+$). A language only needs to be precomputed once and shared among all speaking robots. It can then be used by them for any candidate task.

## IV. Evaluation Results

We refer to the approximate solution in Alg. 1 as $Approx$ and the tractable solution in Sec. III-G as $T$-$Approx$. Our evaluation scenarios are in a gridworld-based setting, similar to the running example. At any time step, the robots can move to any adjacent cell or stay. The robots are not allowed to stay in the same cell or swap locations at the same time step to avoid collisions. The candidate tasks ($\mathcal{T}$) for $Approx$ include all possible tasks in the domain unless stated otherwise. Simulations were created using Webots.

### A. Interpreting the Generated Languages

First, we inspect the language in a relatively simple environment in Fig. 6 (right). We assume that the robots can only observe each other in adjacent cells so miscoordinations can lead to collisions. We implemented a brute-force method that exactly computed the minimal language for $D_{MinW^+}$ (Theorem 2). The abstraction returned by such a method contains 3 abstract states with two of them containing only a single ground state in Fig. 6 (right).

To understand the language, first, observe that miscoordinations can only occur in tasks where the robots must swap their locations while starting non-adjacent to each other. In these tasks, miscoordinations can occur where the robots both choose to start first. For all the other tasks, the plan is always unique such that both robots would choose the same plan so no coordination is needed. Hence, a coordination language needs only to specify which robot starts first.

Now, consider the 3-word language above: two words in Fig. 6 (right) denoted by '0' and '1', respectively, with the third denoted by '2'. Consider a scenario where robot $A$ starts at the top middle and $B$ bottom left, and must swap locations. If robot $A$ is to move first, either robot can communicate "202"; similarly, for $B$ to move first, "212" can be communicated. Since the two plan sketches are expressed differently in the coordination language, miscoordination is avoided. What is interesting here is that the "semantic meanings" of abstract states or words are *task context dependent*. For example, depending on the task, '0' may be used in different sentences to express either $A$ or $B$ moving first.

| | env. size | $\lvert\mathcal{T}\rvert$ | $\lvert S\rvert$ | $\lvert Z\rvert$ | time (s) | $\lvert\approx Z\rvert$ | time (s) | $\lvert\mathcal{T}_5\rvert$ | $\lvert Z_5\rvert$ | time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| GW #1 | $2 \times 2$ | 132 | 12 | 3 | 8.3 | **4** | 0.1 | 68 | 4 | **0.1** |
| GW #2 | $2 \times 3$ | 870 | 30 | — | $\gg$ 36000.0 | **10** | 0.5 | 438 | 10 | **0.4** |
| GW #3 | $2 \times 4$ | 3080 | 56 | — | $\gg$ 36000.0 | **12** | 9.0 | 1544 | 12 | **4.1** |
| GW #4 | $3 \times 3$ | 5112 | 72 | — | $\gg$ 36000.0 | **17** | 43.5 | 2552 | 17 | **18.0** |
| | env. size | $\lvert\mathcal{T}\rvert$ | $\lvert S\rvert$ | $Manhattan(I,G)$ | | $\approx Z$ | time (s) | $\lvert\mathcal{T}_5\rvert$ | $\lvert Z_5\rvert$ | time (s) |
| GWB #1 | $2 \times 4$ | 3080 | 56 | all possible tasks | | **12** | 186.5 | 2920 | 12 | **15.9** |
| GWB #2 | $3 \times 3$ | 3080 | 56 | all possible tasks | | **11** | 211.6 | 2920 | 11 | **14.8** |
| GWB #3 | $3 \times 4$ | 8010 | 90 | all possible tasks | | **13** | 71714.8 | 6330 | 4 | **47.9** |
| GWB #4 | $2 \times 5$ | 8010 | 90 | all possible tasks | | **13** | 112129.1 | 6330 | 4 | **55.6** |
| GWB #5 | $3 \times 3$ | 380 | 56 | $\geq 4$ | | **5** | 1.4 | | | |
| GWB #6 | $3 \times 4$ | 636 | 90 | $\geq 5$ | | **4** | 13.5 | | | |
| GWB #7 | $3 \times 5$ | 956 | 132 | $\geq 6$ | | **4** | 191.4 | | | |
| GWB #8 | $4 \times 4$ | 956 | 132 | $\geq 6$ | | **4** | 199.6 | | | |

TABLE I: Performance of both approximate solutions implementing Alg. 1 (with $\mathcal{T}$ and $\mathcal{T}_5$, respectively) compared to a brute-force solution for $D_{MinW^+}$ in a multi-robot pathfinding domain. Column '$|Z|$' is the vocabulary size returned by brute-force and '$|\approx Z|$' and '$|Z_5|$' are the sizes returned by the approximate solutions, respectively. The times for computing plans for all candidate tasks are included for the approximate solutions.

### B. Language Properties

In this evaluation, we implemented $Approx$ (using $\mathcal{T}$) and $T$-$Approx$ (using $\mathcal{T}_k$), and compared them with the brute-force method discussed in Sec. IV-A. For the approximate solutions, we considered only $\alpha^*$ in Theorem 3 and implemented a greedy algorithm to determine the minimal state abstraction (for finding $f^+$), i.e., adding any remaining states to an abstract state until no more can be added.

**Language Construction.** Table I shows the results. The top part is with open grid-worlds, and the bottom with grid-worlds traversable only at the border cells (GWB): all inner cells are untraversable. We can see that both of our approximate solutions are effective at finding coordination languages with a small vocabulary that partitions the state space. In many cases, our approximate solutions returned a vocabulary size that was significantly smaller than the set of ground actions (25 since each robot has 5 different actions). We can observe that restricting the plan length in $T$-$Approx$ does not have much influence on the vocabulary size when the environments are small (compared to $Approx$); however, for larger environments (with longer plans), the influence is substantial (GWB #3 and #4). We analyze its impact on coordination in Sec. IV-C. We also analyzed restricting to a subset of candidate tasks of $\mathcal{T}$ in $Approx$ whose plans were guaranteed to be longer than a threshold (in contrast to a maximum length restriction in $T$-$Approx$). Towards this end, we evaluated $Approx$ in environments (GWB #5-8) where the Manhattan distances between $I$ and $G$ for one of the robots must be no less than a specified threshold $m$. We observe that restricting $\mathcal{T}$ in $Approx$ in such a way have a similar impact on the vocabulary size as in $T$-$Approx$. We also observe that the computational saving in $T$-$Approx$ (as compared to $Approx$) due to its polynomial requirement is significant (GWB #1-4). Finally, we observe that the vocabulary size stays invariant across some of the environments, implying commonalities among the languages that are independent of the environment size.

Next, we analyze the proposed language in its most restrictive form as computed by $Approx$. The benefits with the language by $T$-$Approx$ would be more substantial (at the cost of losing RC-freeness analyzed in Sec. IV-C).

**Communication Saving.** We compared with a baseline that

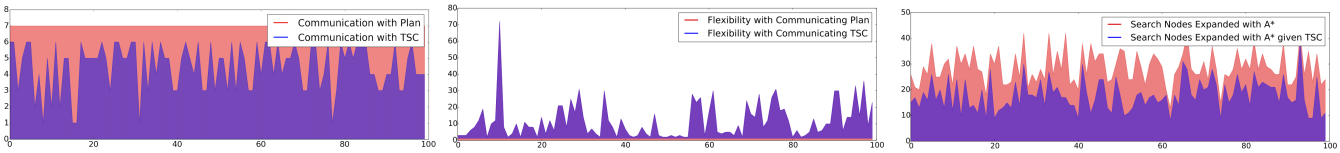


Fig. 7: The X-axis indexes into the problems selected for plotting (with details explained in the text). **Left**: Communication cost: plans vs. TSCs for GWB #7 in Table I, where the state sequence length is always 7. **Middle**: Execution flexibility: plans vs. TSCs for GWB #7. **Right**: Planning cost: A* vs. A* with TSC for GWB #7.

| Env Size | Success Rate w/o replanning | Success Rate with Replanning \| Collision | | | | | |
|---|---|---|---|---|---|---|---|
| | | *Approx* | | *T-Approx* with $k=2$ | | Random | |
| $3\times3$ | 28.4% | +1.1% | 0% | **+15.2%** | 2.3% | +40.9% | **30.7%** |
| $4\times3$ | 29.1% | +1.4% | 0% | **+18.5%** | 1.0% | +41.6% | **29.3%** |
| $5\times3$ | 31.1% | +0.7% | 0% | **+23.5%** | 3.6% | +37.9% | **31.0%** |
| $4\times4$ | 36.1% | +1.1% | 0% | **+44.6%** | 9.4% | +37.1% | **26.8%** |

Fig. 8: **Left**: Problem setting for the navigation domain with dynamic obstacles (i.e., a human worker). **Right**: The success rates (reported as absolute changes with respect to the success rate of the baseline without replanning) and collision rates of 1000 tasks with dynamic obstacles for all methods.

communicates the entire plan. The communication cost of the baseline is determined by the length of the plan, and for *Approx* the length of the TSC. Given a plan chosen by the communicating agent, a sentence is obtained by translating it to the corresponding TSC, which is unique. Even though some languages in prior work [51], [52] also have this feature, translating in the other direction is difficult (see Sec. IV-B) due to incomplete plan specification [53]. We chose the setting for GWB #7 in Table I since the effect of abstraction was more prominent. The result is in Fig. 7 (left) for only the first $100/784$ problems where the communication cost differs. The average saving is **33.3%** among the 784 plans (**27.3%** among all the 956 plans).

**Increased Flexibility.** To verify that TSC increases execution flexibility, using the same setting as above, we evaluated the number of plans that were available to a robot (e.g., listener) after receiving a TSC from the other robot (i.e., speaker). In this evaluation, the speaker always chose the first plan that was found and translated it to a TSC in the coordination language. The result is in Fig. 7 (middle) for the first $100/774$ problems where the number of available plans with TSC is more than one. The increase in flexibility is notable: the average number of plans is **14.7** among the 774 plans or **12.1** among all the 956 plans (compared to 1 with the baseline).

**Reduced Planning Cost.** We evaluated here how coordination could help the listener receiving a TSC reduce its planning cost by providing "*planning guidance*", compared with planning without such guidance. An A* search was implemented with Manhattan distance as the heuristic. We also modified the A* search to consider a given TSC (sent by the speaker). More specifically, when expanding a node, the modified algorithm would not consider its neighbors that were not aligned with the given TSC. The result is presented in Fig. 7 (right) for the first 100 out of all the 956 problems. The average node reduction is **1.6**-fold among all the 956 problems, which is substantial. For the languages studied in [51], [52], the listener would have to compute all the plans and identify one compatible with the language expression, which is more expensive than computing a plan itself!

### C. Robust Navigation with Dynamic Obstacles

Here, we demonstrate how the language contributes to robust coordination. We consider a warehouse setting (see Fig. 8 (left)) where robots are tasked to deliver products between one of the storage zones (located at the corners of the workspace and labeled as $S1$ and $S2$) and one of the dispatch zones (located at the other corners of the workspace and labeled as $D1$ and $D2$). Products must be transported between the corresponding zones (i.e., $S1-D1$ and $S2-D2$). For a given task, the robots start randomly from different corners and must deliver, respectively, to the corresponding zones for storage or dispatch. At the same time, a human worker may be present in the workspace at a random location other than the four corners. We assume that the human worker would not change his/her location during the task. Since the robots are from different manufacturers, they would not be able to robustly detect each other but can both detect the human. To guarantee safety, the robots must coordinate to avoid collisions with each other and the human. We assume that robots move at the same speed. The robots can coordinate their plans via a coordination language before execution but can only detect the position of the human *after* the plan execution starts.

We tested the success rates of 1000 randomly generated tasks when the robots used the exact plan or a TSC for expressing the plan to coordinate using the language computed by *Approx* and *T-Approx* with $k=2$. When the coordination language is used, the robots can choose other candidate plans (if available) that are expressed by the TSC even when the initial plan would lead to a collision with the human. Fig. 8 (right) shows the results for environments of different sizes where a language is constructed for each environment. We can see that although the language computed by *Approx* did not lead to any collisions, it only marginally increased the success rate as a result of the additional flexibility. The language computed by *T-Approx*, on the other hand, provided substantially more flexibility while introducing a small amount of collisions. The baseline where the robots randomly re-pick a plan that would not collide with the human, thus providing the maximum flexibility since an alternative plan always exists in these environments, led to a significantly increased collision risk. Hence, the language computed by *T-Approx* that trades off flexibility with collision risk could be more useful in practice.

## V. Conclusions and Discussions

A novel problem for forming Agent Coordination Language (ACoL) was introduced. We viewed language as a machinery for resolving miscoordinations and reverse-

engineered a language to maximize flexibility during plan execution while guaranteeing optimality. We formally studied the language formation problem and showed that it is NEXP-complete. Two approximate solutions were then developed and evaluated comprehensively. Future work includes relaxing the requirement of plan optimality and/or RC-freeness for more flexibility, coordinating multiple agents (using the language constructed for two agents or, more generally, constructing languages for multiple agents), relaxing the knowledge assumption of the full joint domain model, and incorporating observations. Application-wise, robots communicating "*plan sketches*" may also be used to conserve privacy [54]. The flexibility in choosing plans may alternatively be interpreted as hiding information (e.g., plan preferences) from other agents. Another possible application is to consider communication denied environments where the language can be designed to handle situations where certain parts of the environment are prone to communication challenges.

**ACKNOWLEDGMENT**: This research is supported in part by the NSF grant 2047186. The author would also like to thank Li Wang during the initial formation of the idea and anonymous reviewers for their valuable feedback.

## REFERENCES

[1] M. Barbuceanu and M. S. Fox, "Cool: A language for describing coordination in multi agent systems." in *ICMAS*, 1995, pp. 17–24.
[2] P. Xuan, V. Lesser, and S. Zilberstein, "Communication decisions in multi-agent cooperation: Model and experiments," in *AAMAS*, 2001.
[3] T. Finin, R. Fritzson, D. McKay, and R. McEntire, "Kqml as an agent communication language," in *CIKM*. ACM, 1994, pp. 456–463.
[4] S. Poslad, "Specifying protocols for multi-agent systems interaction," *TAAS*, vol. 2, no. 4, p. 15, 2007.
[5] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *AIJ*, 2015.
[6] R. Nissim, R. I. Brafman, and C. Domshlak, "A general, fully distributed multi-agent planning algorithm," in *AAMAS*, 2010.
[7] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.
[8] Z. Bnaya and A. Felner, "Conflict-oriented windowed hierarchical cooperative A*," in *ICRA*. IEEE, 2014.
[9] Y. Zhang, K. Kim, and G. Fainekos, "DISCOF: Cooperative pathfinding in distributed systems with limited sensing and communication range," in *DARS*, 2016.
[10] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE transactions on robotics and automation*, vol. 14, no. 2, pp. 220–240, 1998.
[11] T. Arai, E. Pagello, L. E. Parker, *et al.*, "Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.
[12] R. Nir, A. Shleyfman, and E. Karpas, "Automated synthesis of social laws in strips," in *AAAI*, vol. 34, no. 06, 2020, pp. 9941–9948.
[13] M. Fox, A. Gerevini, D. Long, and I. Serina, "Plan stability: Replanning versus plan repair." in *ICAPs*, vol. 6, 2006, pp. 212–221.
[14] Y. Zhang, "From abstractions to grounded languages for robust coordination of task planning robots," in *AAMAS*, 2023, pp. 2535–2537.
[15] L. Steels, "Evolving grounded communication for robots," *Trends in cognitive sciences*, vol. 7, no. 7, pp. 308–312, 2003.
[16] M. H. Christiansen and S. Kirby, *Language evolution*. OUP Oxford, 2003.
[17] A. Cangelosi and D. Parisi, *Simulating the evolution of language*. Springer Science & Business Media, 2012.
[18] L. Steels, "Grounding language through evolutionary language games," in *Language Grounding in Robots*. Springer, 2012, pp. 1–22.
[19] S. Kirby, T. Griffiths, and K. Smith, "Iterated learning and the evolution of language," *Current opinion in neurobiology*, 2014.
[20] M. Steedman, "Plans, affordances, and combinatory grammar," *Linguistics and Philosophy*, vol. 25, no. 5-6, pp. 723–753, 2002.
[21] N. Allott, *Game Theory and Communication*. London: Palgrave Macmillan UK, 2006, pp. 123–152.
[22] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 335–346, 1990.
[23] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Approaching the symbol grounding problem with probabilistic graphical models," *AI magazine*, 2011.
[24] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy, "Asking for help using inverse semantics," in *RSS*, Berkeley, USA, July 2014.
[25] Z. Gong and Y. Zhang, "Temporal spatial inverse semantics for robots communicating with humans," in *ICRA*. IEEE, 2018.
[26] S. Kambhampati, C. A. Knoblock, and Q. Yang, "Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning," *Artificial Intelligence*, 1995.
[27] J. Kvarnström and P. Doherty, "Talplanner: A temporal logic based forward chaining planner," *Annals of mathematics and Artificial Intelligence*, vol. 30, no. 1-4, pp. 119–169, 2000.
[28] K. Erol, J. Hendler, and D. S. Nau, "HTN planning: Complexity and expressivity," in *AAAI*, 1994, pp. 1123–1128.
[29] B. J. Clement, E. H. Durfee, and A. C. Barrett, "Abstract reasoning for planning and coordination," *JAIR*, vol. 28, pp. 453–515, 2007.
[30] F. Oliehoek, S. Witwicki, and L. Kaelbling, "A sufficient statistic for influence in structured multiagent environments," *JAIR*, 2021.
[31] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *AIJ*, vol. 112, no. 1, pp. 181 – 211, 1999.
[32] M. Y. Vardi, "An automata-theoretic approach to linear temporal logic," in *Logics for concurrency*. Springer, 1996, pp. 238–266.
[33] E. A. Emerson, "Temporal and modal logic," in *Formal Models and Semantics*. Elsevier, 1990, pp. 995–1072.
[34] D. Abel, D. E. Hershkowitz, and M. L. Littman, "Near optimal behavior via approximate state abstraction," *arXiv preprint arXiv:1701.04113*, 2017.
[35] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Constructing symbolic representations for high-level planning," in *AAAI*, 2014.
[36] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *IJRR*, 2012.
[37] S. Bansal, K. S. Namjoshi, and Y. Sa'ar, "Synthesis of coordination programs from linear temporal specifications," *Proceedings of the ACM on Programming Languages*, vol. 4, no. POPL, pp. 1–27, 2019.
[38] S. Sukhbaatar, R. Fergus, *et al.*, "Learning multiagent communication with backpropagation," in *NIPS*, 2016, pp. 2244–2252.
[39] C. V. Goldman and S. Zilberstein, "Optimizing information exchange in cooperative multi-agent systems," in *AAMAS*, 2003.
[40] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *AAAI*, 2018.
[41] J. Andreas, A. Dragan, and D. Klein, "Translating neuralese," *arXiv preprint arXiv:1704.06960*, 2017.
[42] Y. Zhang, "From abstractions to grounded languages for robust coordination of task planning robots," *CoRR*, vol. abs/1905.00517, 2024. [Online]. Available: http://arxiv.org/abs/1905.00517
[43] P. J. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings," *JAIR*, 2005.
[44] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *AIJ*, 1971.
[45] N. Chomsky, "Three factors in language design," *Linguistic inquiry*, vol. 36, no. 1, pp. 1–22, 2005.
[46] ——, *The minimalist program*. MIT press, 2014.
[47] L. Steels, "Modeling the cultural evolution of language," *Physics of life reviews*, vol. 8, no. 4, pp. 339–356, 2011.
[48] R. Dechter, N. Flerova, and R. Marinescu, "Search algorithms for m best solutions for graphical models," in *AAAI*, 2012.
[49] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.
[50] D. Gentner and L. L. Namy, "Analogical processes in language learning," *Current Directions in Psychological Science*, 2006.
[51] L. Wang and Q. Guo, "Coordination of multiple autonomous agents using naturally generated languages in task planning," *Applied Sciences*, vol. 9, no. 17, p. 3571, 2019.
[52] ——, "Generating the minimal optimal language for cooperative agents," *IEEE Access*, vol. 7, 2019.
[53] M. L. Ginsberg, "Approximate planning," *Artificial Intelligence*, vol. 76, no. 1, pp. 89 – 123, 1995, planning and Scheduling.
[54] R. I. Brafman, "A privacy preserving algorithm for multi-agent planning and search," in *IJCAI*, 2015.