Optimal Scene Graph Planning with Large Language Model Guidance

Zhirui Dai[1] Arash Asgharivaskasi[1] Thai Duong[1] Shusen Lin[1] Maria-Elizabeth Tzes[2] George Pappas[2] Nikolay Atanasov[1]

***Abstract*— Recent advances in metric, semantic, and topological mapping have equipped autonomous robots with concept grounding capabilities to interpret natural language tasks. Leveraging these capabilities, this work develops an efficient task planning algorithm for hierarchical metric-semantic models. We consider a scene graph model of the environment and utilize a large language model (LLM) to convert a natural language task into a linear temporal logic (LTL) automaton. Our main contribution is to enable optimal hierarchical LTL planning with LLM guidance over scene graphs. To achieve efficiency, we construct a hierarchical planning domain that captures the attributes and connectivity of the scene graph and the task automaton, and provide semantic guidance via an LLM heuristic function. To guarantee optimality, we design an LTL heuristic function that is provably consistent and supplements the potentially inadmissible LLM guidance in multi-heuristic planning. We demonstrate efficient planning of complex natural language tasks in scene graphs of virtualized real environments.**

I. INTRODUCTION

Advances in robot perception and computer vision have enabled metric-semantic mapping [1]–[9], offering rich information in support of robot autonomy. Beyond single-level maps, hierarchical models encode topological relations among local maps and semantic elements [10], [11]. A scene graph [11] is a prominent example that models buildings, floors, rooms, objects, and occupancy in a unified hierarchical representation. Scene graph construction can be done from streaming sensor data [5], [12], [13]. The metric, semantic, and topological elements of such models offer the building blocks for robots to execute semantic tasks [14]. The objective of this work is to approach this challenge by generalizing goal-directed motion planning in flat geometric maps to natural language task planning in scene graphs.

Connecting the concepts in a natural language task to the real-world objects they refer to is a challenging problem, known as symbol grounding [15]. Large language models (LLMs), such as GPT-3 [16], BERT [17], and LLaMA [18], offer a possible resolution with their ability to relate environment entities to concepts in natural language. Chen et al. [19], [20] use LLMs for scene graph labeling, showing their capability of high-level understanding of indoor scenes. Shah et al. [21] use GPT-3 to parse text instructions to landmarks and the contrastive language image pre-training model [22] to infer a joint landmark-image distribution for visual navigation. Seminal papers [23], [24] in the early 2000s established formal logics and automata as powerful representations of robot tasks. In works closely related to ours, Chen et al. [25] show that natural language tasks in 2D maps encoded as sets of landmarks can be converted to signal temporal logic [26] via LLM re-prompting and automatic syntax correction, enabling the use of existing temporal logic planners [27]–[31]. Beyond temporal logics, other expressive robot task representations include the planning domain definition language [32], Petri nets [33], [34], and process algebra [35]. Oh et al. [36] train a neural model to translate a natural language task to LTL and use abstract product Markov decision process to solve hierarchical planning problem considering accepting LTL traces. We use an LLM to translate a natural language task grounded to semantic attributes of a scene graph to a linear temporal logic (LTL) formula [37]. We describe the scene graph to the LLM as an attribute hierarchy and obtain task execution guidance from the LLM to accelerate the planning algorithm. We employ multi-heuristic planning to achieve both efficiency due to semantic LLM guidance and optimality due to consistent LTL guidance.

Given a symbolically grounded task representation, the next challenge is to plan its execution in a hierarchical model efficiently with optimality guarantee. A key component for achieving efficiency in traditional goal-oriented planning is the use of guidance from a heuristic function. Heuristics play a critical role in accelerating both search-based [40], [41] and sampling-based [42], [43] planners. More complex tasks than goal reaching involve sequencing, branching, and recurrence, making heuristic guidance even more important for efficiency. Our work is inspired by Fu et al. [29] who develop an admissible heuristic for LTL planning in probabilistic landmark maps. We extend the approach by (a) generalizing to hierarchical scene graphs via multi-resolution planning, (b) designing a consistent LTL heuristic allowing acceleration over admissible-only heuristic planning, and (c) introducing an LLM heuristic allowing acceleration from LLM semantic guidance while retaining *optimality guarantees* via multi-heuristic planning. Our approach is enabled by the anytime multi-resolution multi-heuristic A* (AMRA*) [39] . Our key contribution is to define the nodes, edges, and

We gratefully acknowledge support from ARL DCIST CRA W911NF-17-2-0181 and ONR N00014-23-1-2353.

[1]The authors are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA, e-mails: {zhdai,shl090,tduong,aasghari, natanasov}@ucsd.edu.

[2]The authors are with GRASP Lab, University of Pennsylvania, Philadelphia, PA 19104, USA, e-mails: {mtzes,pappasg} @seas.upenn.edu.

Implementation available at www.existentialrobotics.org/pages/llm-planning.html.

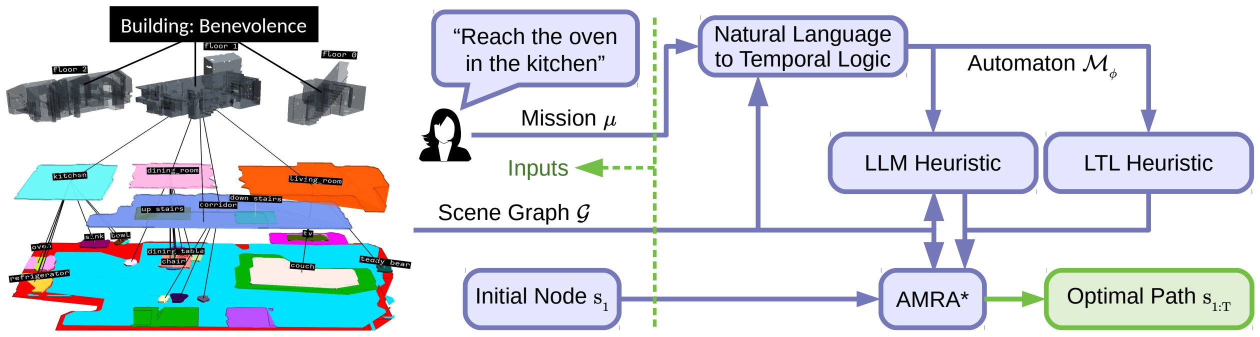


Fig. 1: Planning a natural language mission, μ : "Reach the oven in the kitchen", in a scene graph $\mathcal{G}$ of the Gibson environment Benevolence [38] with object, room, and floor attributes. The terms "oven" and "kitchen" in μ belong to the object and room attributes of the scene graph, respectively. The scene graph $\mathcal{G}$ is described to the LLM using the connectivity of its attributes (attribute hierarchy) and the LLM is used to translate μ to LTL formula ϕ_μ and associated Automaton $\mathcal{M}_\phi$. We construct a hierarchical planning domain from the scene graph, and use multi-resolution multi-heuristic planning [39] to plan the mission execution. In addition to mission translation, the LLM is used to provide heuristic guidance to accelerate the planning, while an LTL heuristic is used to guarantee optimality.

costs of a hierarchical planning domain from a scene graph and to introduce guidance from a consistent LTL heuristic and a semantically informed LLM heuristic.

Related works consider object search and semantic goal navigation in unknown environments, represented as semantic occupancy maps [44], topological maps [45], or scene graphs [46], [47]. Shah et al. [44] develop an exploration approach that uses an LLM to score subgoal candidates and provide an exploration heuristic. Kostavelis et al. [45] perform place recognition using spatial and temporal proximity to obtain a topological map and encode the connectivity of its attributes in a navigation graph to enable Dijkstra planning to semantically specified goals. Amiri et al. [46] employ a scene graph generation network to construct a scene graph from images and a partially observable Markov decision process to obtain an object-search policy. Ravichandran et al. [47] embed partial scene graph observations in feature space using a graph neural network (GNN) and train value and policy GNNs via proximal policy optimization. Liang et al. [48] develop Code as Policies, an approach for LLM code synthesis that utilizes action primitive APIs to solve object manipulation tasks. Ding et al. [49] use an LLM to create both symbolic and geometric spatial relationships among tableware objects to assist task and motion planning. In contrast with these works, we consider significantly more general tasks but perform planning in a known scene graph.

In summary, this paper makes the following *contributions*.

- We use an LLM to translate natural language to LTL tasks over the attributes of a scene graph.
- We construct a hierarchical planning domain capturing the structure of the scene graph and LTL task.
- We design new LTL and LLM heuristic functions for planning, and prove that the LTL heuristic is consistent.
- We employ hierarchical multi-heuristic planning to guarantee efficiency (due to LLM semantic guidance) and optimality (due to LTL consistent guidance), despite potential inadmissibility of the LLM heuristic.

II. PROBLEM STATEMENT

Consider an agent planning a navigation mission specified in terms of semantic concepts, such as objects and regions, in a known environment. We assume that the environment is represented as a scene graph, e.g., obtained from [11].

Definition 1. A *scene graph* is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{\mathcal{A}_k\}_{k=1}^K)$ with node set $\mathcal{V}$, edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and K attribute sets $\mathcal{A}_k$ for $k = 1, \ldots, K$. Each attribute $a \in \mathcal{A}_k$ is associated with a subset $\mathcal{V}_a$ of the nodes $\mathcal{V}$.

A scene graph provides a hierarchical semantic description of an environment, such as a building, in terms of floors, rooms, objects, and occupancy (see Fig. 1). For example, the graph nodes $\mathcal{V}$ may represent free space, while the objects may be encoded as an attribute set $\mathcal{A}_1$ such that an object $o \in \mathcal{A}_1$ is associated with a free region $\mathcal{V}_o \subset \mathcal{V}$ around it. Similarly, rooms may be represented as a set $\mathcal{A}_2$ such that a room $r \in \mathcal{A}_2$ is associated with a free-space region $\mathcal{V}_r \subset \mathcal{V}$.

Given the initial agent location $s \in \mathcal{V}$, and a cost function $c : \mathcal{E} \mapsto \mathbb{R}_{>0}$, our objective is to plan a sequence of scene graph nodes (path) that achieves a mission μ with minimal cost. We assume μ is provided in natural language using terms from the attribute sets $\mathcal{A}_k$ of the scene graph. An example scene graph mission is provided in Fig. 1.

To interpret a natural language mission, we define atomic propositions whose combinations can capture the mission requirements. An *atomic proposition* $p_a : \mathcal{V} \mapsto \{\texttt{false}, \texttt{true}\}$ associated with attribute $a \in \mathcal{A}_k$ of the scene graph $\mathcal{G}$ evaluates true at node $s \in \mathcal{V}$ if s belongs to the nodes $\mathcal{V}_a$ that satisfy attribute a. We denote this by $p_a(s) : s \in \mathcal{V}_a$. The set of all atomic propositions at $s \in \mathcal{V}$ is denoted by:

$$\mathcal{AP}(s) := \{p_a(s) \mid a \in \mathcal{A}_k, k = 1, \ldots, K\}. \tag{1}$$

Intuitively, $p_a(s)$ being true means that the agent is at node s that satisfies attribute a, e.g., reaching an object in $\mathcal{A}_1$ or being in a room in $\mathcal{A}_2$. Avoiding an object or leaving a room can be specified via the negation of such propositions. To determine which atomic propositions are satisfied at a particular node, we define a labeling function.

Definition 2. Consider a scene graph $\mathcal{G}$ with atomic propositions $\mathcal{AP} = \cup_{s \in \mathcal{V}} \mathcal{AP}(s)$. A *labeling function* $\ell : \mathcal{V} \mapsto 2^{\mathcal{AP}}$ maps a node $s \in \mathcal{V}$ to a set $\ell(s) \subseteq \mathcal{AP}$ of atomic propositions that evaluate $\texttt{true}$ at s.

The labels along a path $s_{1:T}$ are obtained as $\ell(s_{1:T}) = \ell(s_1)\ell(s_2)\dots\ell(s_T)$ and are called a *word*. A word contains information about the objects, rooms, and floors that an agent visits along its path in a scene graph. We can decide whether the agent path satisfies a mission μ by interpreting its word. We denote a word $\ell(s_{1:T})$ that satisfies a mission μ by $\ell(s_{1:T}) \models \mu$, and define the precise semantics of this notation in Sec. III. With this, we are ready to state the problem of natural language mission planning in scene graphs.

Problem. Given a scene graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{\mathcal{A}_k\}_{k=1}^K)$, natural language mission μ over the attributes of $\mathcal{G}$, cost function $c : \mathcal{E} \mapsto \mathbb{R}_{>0}$, and initial node $s_1 \in \mathcal{V}$, plan a path $s_{1:T}$ that satisfies μ with minimal cost:

$$\begin{aligned} \min_{T\in\mathbb{N}, s_{1:T}\in\mathcal{V}^T} & \sum_{t=1}^{T-1} c(s_t, s_{t+1}) \\ \text{s.t.} \quad & (s_t, s_{t+1}) \in \mathcal{E}, \quad t = 1, \dots, T-1, \\ & \ell(s_{1:T}) \models \mu. \end{aligned} \tag{2}$$

III. NATURAL LANGUAGE TO TEMPORAL LOGIC

We use an LLM to translate NL missions to LTL over the scene graph propositions $\mathcal{AP}$. The key challenge is to describe a scene graph $\mathcal{G}$ to an LLM and ask it to translate a mission μ to an LTL formula ϕ_μ. We focus on LTL [50] with syntax in Table. I due to its popularity and sufficient expressiveness to capture temporal ordering of propositions.

We require a syntactically co-safe formula (sc-LTL) [37] to allow checking its satisfaction over finite paths. An sc-LTL can be satisfied by a word $\ell(s_{1:T})$ that consists of a finite prefix followed by a potentially infinite continuation that does not affect the formula's truth value. LTL formulas that contain only $\mathbf{X}$, $\mathbf{F}$ and $\mathbf{U}$ temporal operators when written in positive normal form ($\neg$ appears only in front of atomic propositions) are syntactically co-safe [36], [37].

To use an LLM for scene understanding, it is necessary to design a prompt that describes the scene graph $\mathcal{G}$ succinctly. Otherwise, the input might exceed the model token limit (e.g., GPT-4 [51] has a token limit of 8192) or confuse the model about the relationship between the sentences. For this aim, we simplify the scene graph $\mathcal{G}$ into an *attribute hierarchy* $\bar{\mathcal{G}}$ that compactly represents scene entities in a YAML format. In our setup, the top level of $\bar{\mathcal{G}}$ contains floors $f \in \mathcal{A}_3$. The rooms r_f on floor f are defined as $\{r \in \mathcal{A}_2 | \mathcal{V}_r \subseteq \mathcal{V}_f\}$, and nested as children of floor f. Additionally, each room r stores connections to other rooms on the same floor. Similarly, the objects in room r, $\{o \in \mathcal{A}_1 | \mathcal{V}_o \subseteq \mathcal{V}_r\}$, are stored as children of room r. Each entity in $\bar{\mathcal{G}}$ is tagged with a unique ID to differentiate rooms and objects with the same name. See Fig. 2a for an example attribute hierarchy. In our examples, we define attributes for floors, rooms, and objects. The attribute hierarchy contains 3 levels but this can be extended to more generalized attribute sets (e.g., level for object affordances). The attribute hierarchy removes the dense node and edge descriptions in $\mathcal{G}$ which are redundant for mission translation, leading to a significant reduction in prompt size.

TABLE I: Grammar for LTL formulas ϕ and φ.

Syntax (Semantics)					
p_a	(Atomic Proposition)	$\phi \vee \varphi$	(Or)	$\phi \mathbf{U} \varphi$	(Until)
$\neg\phi$	(Negation)	$\phi \Rightarrow \varphi$	(Imply)	$\mathbf{F}\phi$	(Eventually)
$\phi \wedge \varphi$	(And)	$\mathbf{X}\phi$	(Next)	$\mathbf{G}\phi$	(Always)

Given the natural language mission μ and the attribute hierarchy $\bar{\mathcal{G}}$, we first call the LLM to extract unique IDs from the context of μ, outputting μ_{unique}. This step facilitates LTL translations by separating high-level scene understanding from accurate LTL generation. Specifically, this step links contextually rich specifications to unequivocal mission descriptions that are void of confusion for the LLM (Fig. 2b). The list of entities involved in the mission are extracted from μ_{unique} using regular expression, and stored as μ_{regex}. This allows to inform the LLM about the essential parts of the mission μ_{unique}, without providing $\bar{\mathcal{G}}$. The savings in prompt size are used to augment the prompt with natural language to LTL translation examples expressed in *pre-fix* notation. For instance, $\phi \wedge \varphi$ is expressed as $\wedge\phi\varphi$ in pre-fix format. This circumvents the issue of balancing parenthesis over formula ϕ_μ. Ultimately, the LTL translation prompt includes μ_{unique}, μ_{regex}, and the translation examples (Fig. 2c).

The translated LTL formula ϕ_μ is verified for syntactic correctness and syntactical co-safety using an LTL syntax checker. Further calls of the LLM are made to correct ϕ_μ if it does not pass the checks, up to a maximum number of allowed verification steps, after which human feedback is used to rephrase the natural language specification μ (Fig. 2d). Once the mission μ is successfully translated into an sc-LTL formula ϕ_μ, we can evaluate whether an agent path $s_{1:T}$ and its corresponding word $\ell(s_{1:T})$ satisfy ϕ_μ by constructing an automaton representation of ϕ_μ (Fig. 2e).

Definition 3. A deterministic automaton over atomic propositions $\mathcal{AP}$ is a tuple $\mathcal{M} = (\mathcal{Q}, 2^{\mathcal{AP}}, T, \mathcal{F}, q_1)$, where $\mathcal{Q}$ is a set of states, $2^{\mathcal{AP}}$ is the power set of $\mathcal{AP}$ called alphabet, $T : \mathcal{Q} \times 2^{\mathcal{AP}} \mapsto \mathcal{Q}$ is a transition function that specifies the next state $T(q, l)$ from state $q \in \mathcal{Q}$ under label $l \in 2^{\mathcal{AP}}$, $\mathcal{F} \subseteq \mathcal{Q}$ is a set of final states, and $q_1 \in \mathcal{Q}$ is an initial state.

An sc-LTL formula ϕ can be translated into a deterministic automaton $\mathcal{M}_{\phi_\mu}$ using model checking tools such as Spot [52]. The automaton checks whether a word satisfies ϕ_μ. A word $\ell_{1:T}$ is accepted by $\mathcal{M}_{\phi_\mu}$, i.e., $\ell_{1:T} \models \phi_\mu$, if and only if the state q_{T+1} obtained after transitions $q_{t+1} = T(q_t, \ell_t)$ for $t = 1, \dots, T$ is one of the final states $\mathcal{F}$. Hence, a path $s_{1:T}$ satisfies an sc-LTL formula ϕ_μ if and only if its word $\ell(s_{1:T})$ contains a prefix $\ell(s_{1:t})$ that is accepted by $\mathcal{M}_{\phi_\mu}$.

IV. OPTIMAL SCENE GRAPH PLANNING

We use the structure of the scene graph $\mathcal{G}$ with guidance from the automaton $\mathcal{M}_{\phi_\mu}$ and the LLM's mission semantics understanding to achieve efficient and optimal planning.

A. AMRA* Planning

We use AMRA* [39]. The key challenge is to define a hierarchical planning domain and heuristics that describe the scene graph and mission. AMRA* requires node sets

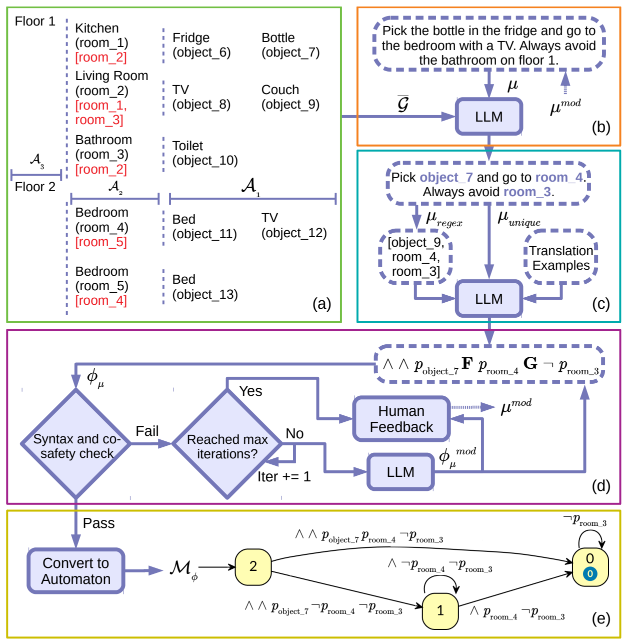

Floor 1
Kitchen (room_1) [room_2]
Living Room (room_2) [room_1, room_3]
Bathroom (room_3) [room_2]
Fridge (object_6)
Bottle (object_7)
TV (object_8)
Couch (object_9)
Toilet (object_10)
$\mathcal{A}_3$
Floor 2
$\mathcal{A}_2$
$\mathcal{A}_1$
Bedroom (room_4) [room_5]
Bed (object_11)
TV (object_12)
Bedroom (room_5) [room_4]
Bed (object_13)
(a)
Pick the bottle in the fridge and go to the bedroom with a TV. Always avoid the bathroom on floor 1.
$\bar{\mathcal{G}}$
μ
μ^{mod}
LLM
(b)
Pick object_7 and go to room_4. Always avoid room_3.
μ_{regex}
μ_{unique}
[object_9, room_4, room_3]
Translation Examples
LLM
(c)
$\wedge \wedge p_{\text{object\_7}}\, \mathbf{F}\, p_{\text{room\_4}}\, \mathbf{G} \neg p_{\text{room\_3}}$
ϕ_μ
Syntax and co-safety check
Fail
Reached max iterations?
Yes
No
Iter += 1
Human Feedback
μ^{mod}
LLM
ϕ_μ^{mod}
(d)
Pass
Convert to Automaton
$\mathcal{M}_\phi$
2
1
0
$\wedge \wedge p_{\text{object\_7}} p_{\text{room\_4}} \neg p_{\text{room\_3}}$
$\wedge \neg p_{\text{room\_4}} \neg p_{\text{room\_3}}$
$\wedge \wedge p_{\text{object\_7}} \neg p_{\text{room\_4}} \neg p_{\text{room\_3}}$
$\wedge\, p_{\text{room\_4}} \neg p_{\text{room\_3}}$
$\neg p_{\text{room\_3}}$
(e)

Fig. 2: Natural language to LTL translation. (a) Attribute hierarchy $\bar{\mathcal{G}}$. The unique IDs and the room connections are shown in parentheses and inside red brackets, respectively. (b) Unique ID extraction from natural language mission μ. (c) LTL formula generation from natural language specification. (d) Syntax and co-safety check over the generated LTL formula ϕ_μ. (e) Automaton construction.

$\mathcal{X}_r$ and action sets $\mathcal{U}_r$ for each planning resolution level, $r = 1, 2, \ldots$. Each level $(\mathcal{X}_r, \mathcal{U}_r)$ has an associated cost function $c_r : \mathcal{X}_r \times \mathcal{X}_r \mapsto \mathbb{R}_{>0}$. The algorithm defines an anchor level $(\mathcal{X}_0, \mathcal{U}_0)$, as $\mathcal{X}_0 := \cup_{r>0}\mathcal{X}_r, \mathcal{U}_0 := \cup_{r>0}\mathcal{U}_r$, and requires an initial state $x \in \mathcal{X}_0$ and a goal region $\mathcal{R} \subseteq \mathcal{X}_0$. AMRA* allows multiple heuristics for each level but requires the anchor-level heuristic to be unique and consistent to guarantee optimality. A heuristic h is consistent with respect to cost c if $h(x) \leq c(x, x') + h(x')$. We construct the levels $\{\mathcal{X}_r, \mathcal{U}_r, c_r\}$, initial state x, and goal region $\mathcal{R}$ required for running AMRA* in Sec. IV-B. We define two heuristics, h_{LTL}, which is used in the anchor level and other levels, and h_{LLM} for other levels, in Sec. IV-C and Sec. IV-D, respectively, and prove that h_{LTL} is consistent.

B. Hierarchical Planning Domain Description

Given a scene graph $\mathcal{G}$ with initial node $s_1 \in \mathcal{V}$ and automaton $\mathcal{M}_{\phi_\mu}$, we construct a hierarchical planning domain with K levels corresponding to each scene graph attribute $\mathcal{A}_k$. Given an attribute set $\mathcal{A}_k$, we define $\mathcal{V}_k := \cup_{a \in \mathcal{A}_k} \mathcal{V}_a$, where $\mathcal{V}_a$ is the node set associated with attribute $a \in \mathcal{A}_k$. Then, the node set corresponding to level k of the planning domain is defined as $\mathcal{X}_k := \mathcal{V}_k \times \mathcal{Q}$. We define the actions $\mathcal{U}_k$ as transitions from x_i to x_j in $\mathcal{X}_k$ with associated cost $c_k(x_i, x_j)$. A transition from $x_i = (s_i, q_i)$ and $x_j = (s_j, q_j)$ exists if the following conditions are satisfied:

1) the transition is from $\mathcal{V}_a$ of attribute a to the boundary $\partial\mathcal{V}_b$ of attribute b such that $s_i \in \mathcal{V}_a$, $s_j \in \partial\mathcal{V}_b$ for $a \neq b$ with $a, b \in \mathcal{A}_k$ and $\text{int}\mathcal{V}_a \cap \text{int}\mathcal{V}_b = \emptyset$,
2) the automaton transitions are respected: $q_j = T(q_i, \ell(s_i))$, where $\ell(s_i) \in 2^{\mathcal{AP}}$ is the label at s_i,

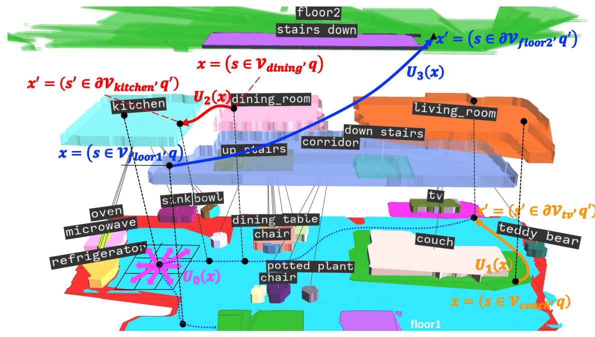

floor2
stairs down
$x' = (s \in \partial\mathcal{V}_{floor2}, q')$
$x = (s \in \mathcal{V}_{dining}, q)$
$x' = (s' \in \partial\mathcal{V}_{kitchen}, q')$
$U_2(x)$
$U_3(x)$
kitchen
dining_room
living_room
down stairs
corridor
$x = (s \in \mathcal{V}_{floor1}, q)$
up stairs
sink bowl
tv
$x' = (s' \in \partial\mathcal{V}_{tv}, q')$
oven
microwave
dining table
chair
teddy bear
refrigerator
couch
$U_0(x)$
potted plant
chair
$U_1(x)$
$x = (s \in \mathcal{V}_{couch}, q)$
floor1

Fig. 3: Four-level hierarchical planning domain for *Benevolence*.

3) the transition is along the shortest path, $s_j \in \arg\min_s d(s_i, s)$, where $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is the shortest feasible path between two scene graph nodes.

The transition cost is defined as $c_k(x_i, x_j) = d(s_i, s_j)$.

Since $\mathcal{X}_k \subseteq \mathcal{V} \times \mathcal{Q}$, we can define the AMRA* anchor level as $\mathcal{X}_0 = \mathcal{V} \times \mathcal{Q}$ with actions $\mathcal{U}_0$ derived from the scene graph edges $\mathcal{E}$, automaton transitions, and $\cup_{k>0}\mathcal{U}_k$. The initial state and goal region are defined as $x = (s_1, q_1)$ and $\mathcal{R} = \mathcal{V} \times \mathcal{F}$.

The hierarchical planning domain is illustrated in Fig. 3. Four levels, occupancy ($\mathcal{V}$), objects ($\mathcal{A}_1$), rooms ($\mathcal{A}_2$) and floors ($\mathcal{A}_3$), are used in our experiments. For example, in the object level, the agent can take an action to move directly from the couch to the TV with transition cost computed as the shortest path in the occupancy level.

C. LTL Heuristic

A consistent heuristic is required at the anchor level to ensure path optimality. Fu et al. [29] designed an LTL heuristic that considers the next label to induce transitions in $\mathcal{M}_{\phi_\mu}$ towards a final state. This heuristic is admissible but not consistent. We extend the formulation to consider the shortest path to reach a final state in the automaton label space. We design a heuristic function h_{LTL} that approximates the scene graph distance to $\mathcal{R} = \mathcal{V} \times \mathcal{F}$. We require that the scene graph cost c satisfies the triangle inequality, $c(s, s') \leq c(s, s'') + c(s'', s')$ for any $s, s', s'' \in \mathcal{V}$. Then, we define the cost between two labels $l_1, l_2 \in 2^{\mathcal{AP}}$ as

$$c_\ell(l_1, l_2) = \min_{s_1, s_2 : \ell(s_1) = l_1, \ell(s_2) = l_2} c(s_1, s_2), \tag{3}$$

which is a lower bound on the transition cost from l_1 to l_2 in the metric space of cost function c. Next, we define a lower bound on the transition cost from automaton state $q \in \mathcal{Q}$ with label $l \in 2^{\mathcal{AP}}$ to an accepting state $q_f \in \mathcal{F}$ as:

$$g(l, q) = \min_{l' \in 2^{\mathcal{AP}}} c_\ell(l, l') + g(l', T(q, l')). \tag{4}$$

The function $g : 2^{\mathcal{AP}} \times \mathcal{Q} \mapsto \mathbb{R}_{\geq 0}$ can be pre-computed via Dijkstra's algorithm on the automaton $\mathcal{M}_{\phi_\mu}$. We also define a next labeling function $\ell_n : 2^{\mathcal{AP}} \times \mathcal{Q} \mapsto 2^{\mathcal{AP}}$ that tracks the least-cost label sequence returned by Dijkstra's algorithm with $g(l, q) = 0$, $\ell_n(l, q) = \texttt{true}$, $\forall q \in \mathcal{F}$.

Proposition 1. *The heuristic function* $h_{\text{LTL}} : \mathcal{V} \times \mathcal{Q} \to \mathbb{R}$

defined below is consistent:

$$\begin{aligned} h_{\text{LTL}}(s,q) &= \min_{t\in\mathcal{V}} \left[c(s,t) + g(l(t), T(q, l(t)))\right], \\ h_{\text{LTL}}(s,q) &= 0, \quad \forall q \in \mathcal{F}. \end{aligned} \tag{5}$$

Proof. Consider a state $x = (s_x, q_x)$ with label $l_x = l(s_x)$. For any $y = (s_y, q_y)$ that s_y is reachable from s_x in one step, we have two cases to handle. When $T(q_x, l(s_y)) = q_y = q_x$:

$$\begin{aligned} h_{\text{LTL}}(s_x, q_x) &\le \min_{t\in\mathcal{V}} \left[c(s_x, s_y) + c(s_y, t) + g(l(t), T(q_x, l(t)))\right] \\ &= c(s_x, s_y) + \min_{t\in\mathcal{V}} \left[c(s_y, t) + g(l(t), T(q_y, l(t)))\right] \\ &= c(s_x, s_y) + h_{\text{LTL}}(s_y, q_y). \end{aligned}$$

When $T(q_x, l_y) = q_y \neq q_x$, with $l_y = l(s_y)$, we have:

$$\begin{aligned} &c(s_x, s_y) + h_{\text{LTL}}(s_y, q_y) \\ &= c(s_x, s_y) + \min_{t\in\mathcal{V}} \left[c(s_y, t) + g(l(t), T(q_y, l(t)))\right] \\ &\ge \min_{t\in\mathcal{V}, l(t)=l_y} c(s_x, t) + \min_{t\in\mathcal{V}} \left[c(s_y, t) + g(l(t), T(q_y, l(t)))\right] \\ &\ge \min_{t\in\mathcal{V}, l(t)=l_y} c(s_x, t) + \min_{l'\in 2^{\mathcal{AP}}} \left[c_l(l_y, l') + g(l', T(q_y, l'))\right] \\ &\ge \min_{t\in\mathcal{V}} \left[c(s_x, t) + g(l(t), T(q_x, l(t)))\right] = h_{\text{LTL}}(s_x, q_x). \quad \square \end{aligned}$$

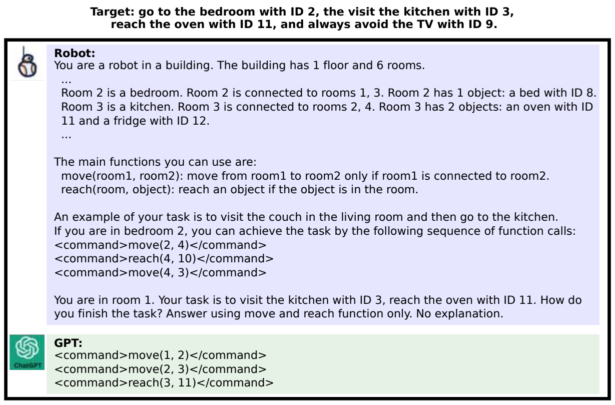
D. LLM Heuristic

In this section, we develop an LLM heuristic $h_{\text{LLM}} : \mathcal{V} \times \mathcal{Q} \to \mathbb{R}$ that captures the hierarchical semantic information of the scene graph. The LLM heuristic uses all attributes at a node $s \in \mathcal{V}$, the current automaton state $q \in \mathcal{Q}$, and the attribute hierarchy $\bar{\mathcal{G}}$, and returns an attribute-based guide that helps the AMRA* to search in the right direction for an optimal path. We design the prompt to ask LLM for attribute-based guidance with 4 components as follows:

- environment description from its attribute hierarchy $\bar{\mathcal{G}}$,
- list of motions $M = \{m_i(\cdot,\cdot)\}$, where $m_i(a_j, a_k)$, $a_j \in \mathcal{A}_j$, $a_k \in \mathcal{A}_k$ describes movements on $\bar{\mathcal{G}}$ from attribute a_i to a_j that the LLM model uses to generate its guides,
- an example of the mission μ_{unique} and how to response,
- description of the mission μ_{unique}, current attributes, remaining task given the automaton state $q \in \mathcal{Q}$, and request for guidance on how to finish the task.

The LLM model returns a sequence of function calls $\{f_i(a_j, a_k)\}_{i=0}^{N}$, $f_i \in M$, $a_j \in \mathcal{A}_j$, $a_k \in \mathcal{A}_k$ in XML format, easing response parsing [53]. Each function call returns a user-defined cost, e.g., Euclidean distance between attributes: $f_i(a_j, a_k) = c(s_j, s_k)$, where s_j, s_k are the center of $\mathcal{V}_{a_j}$ and $\mathcal{V}_{a_k}$, respectively. The total cost of the LLM functions is used as an LLM heuristic $h_{LLM}(s,q) = \sum_{i=0}^{N} f_i(a_j, a_k)$. Due to the LLM query delay and its limited query rates, the sequence of function calls suggested by the LLM model is obtained offline, stored and used to calculate the heuristic h_{LLM} online in AMRA* based on the user-defined cost.

Fig. 4 illustrates a sample prompt and response. The prompt first describes the attribute hierarchy $\bar{\mathcal{G}}$. Each attribute is mentioned with a unique ID to avoid confusing the LLM, as shown in the first paragraph of the prompt in Fig. 4. The second part of the prompt provides a list of possible functions used to guide the agent, such as $move(1,2)$ to move from room 1 to room 2, or $reach(1,3)$ to reach an object 3 in room



Target: go to the bedroom with ID 2, the visit the kitchen with ID 3, reach the oven with ID 11, and always avoid the TV with ID 9.

Robot:
You are a robot in a building. The building has 1 floor and 6 rooms.
...
Room 2 is a bedroom. Room 2 is connected to rooms 1, 3. Room 2 has 1 object: a bed with ID 8.
Room 3 is a kitchen. Room 3 is connected to rooms 2, 4. Room 3 has 2 objects: an oven with ID 11 and a fridge with ID 12.
...

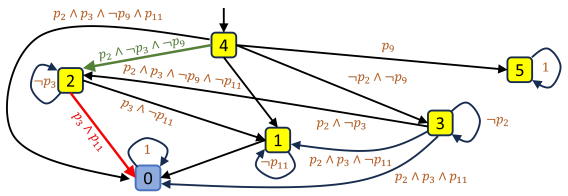
The main functions you can use are:
move(room1, room2): move from room1 to room2 only if room1 is connected to room2.
reach(room, object): reach an object if the object is in the room.

An example of your task is to visit the couch in the living room and then go to the kitchen.
If you are in bedroom 2, you can achieve the task by the following sequence of function calls:
```
<command>move(2, 4)</command>
<command>reach(4, 10)</command>
<command>move(4, 3)</command>
```

You are in room 1. Your task is to visit the kitchen with ID 3, reach the oven with ID 11. How do you finish the task? Answer using move and reach function only. No explanation.

GPT:
```
<command>move(1, 2)</command>
<command>move(2, 3)</command>
<command>reach(3, 11)</command>
```

Fig. 4: ChatGPT prompt requesting a scene graph path.



$p_2 \wedge p_3 \wedge \neg p_9 \wedge p_{11}$
4
$p_2 \wedge \neg p_3 \wedge \neg p_9$
p_9
5
1
$\neg p_3$
2
$p_2 \wedge p_3 \wedge \neg p_9 \wedge \neg p_{11}$
$\neg p_2 \wedge \neg p_9$
$p_3 \wedge \neg p_{11}$
$p_3 \wedge p_{11}$
3
$\neg p_2$
$p_2 \wedge \neg p_3$
1
1
$\neg p_{11}$
$p_2 \wedge p_3 \wedge \neg p_{11}$
0
$p_2 \wedge p_3 \wedge p_{11}$

Fig. 5: The automaton graph T for the mission *"go to the bedroom 2, then visit the kitchen 3, reach the oven 11, and always avoid the TV 9"* with an initial node $q_1 = 4$ and an accepting node 0.

1. The third component provides an example of a mission and how the LLM should response. The last component describes the current attributes and the remaining mission, generated based on the current automaton state q, and requests LLM to generate a high-level plan using the provided functions.

The automaton state q represents how far we have achieved the mission. Thus, to describe the remaining mission, we run Dijkstra's algorithm on the automaton $\mathcal{M}_{\phi_\mu}$ to find the shortest path from q to an accepting state in $\mathcal{F}$. We obtain a set of atomic prepositions evaluated *true* along the path, and concatenate their descriptions to describe the remaining mission in the prompt. For example, the desired mission is to *"go to the bedroom 2, then visit the kitchen 3, reach the oven 11, while always avoid the TV 9"*. Let the atomic prepositions be defined as p_2, p_3, p_{11}, and p_9, where the indices correspond to the ID of the room or object. The task can be described using an LTL as follows: $\phi = \mathbf{F}(p_2 \wedge \mathbf{F}(p_3 \wedge \mathbf{F} p_{11})) \wedge \neg p_9$, whose automaton graph T generated from Spot [52] is shown in Fig. 5 with the initial state $q_1 = 4$ and the final states $\mathcal{F} = \{0\}$. The agent is currently in room 1 and have visited room 2, i.e. $q = 2$. The shortest path from q to the accepting state 0 is marked by red arrows in Fig. 5. Along this path, p_3 and p_{11} turn true, causing the remaining mission to be *"visit the kitchen 3 and reach the oven 11"* (Fig. 4). The atomic preposition p_9 leads to a sink state 5 if it evaluates true, and never appears in the next mission, leading to an optimistic LLM heuristic.

V. EVALUATION

To evaluate our method, we use *Allensville* (1-floor), *Benevolence* (3-floor) and *Collierville* (3-floor) from the 3D Scene Graph dataset [11]. For each scene, we designed 5 missions (some are shown in Table II). For each mission,

TABLE II: Example mission descriptions for each scene.

Allensville	Clean all vases in the dining room. Eventually water the potted plants in the bathrooms one after another.
Benevolence	Visit the bathroom on floor 0 and avoid the sink, then go to the dining room and sit on a chair. Always avoid the living room.
Collierville	Clean all the corridors, except the one on floor 0.

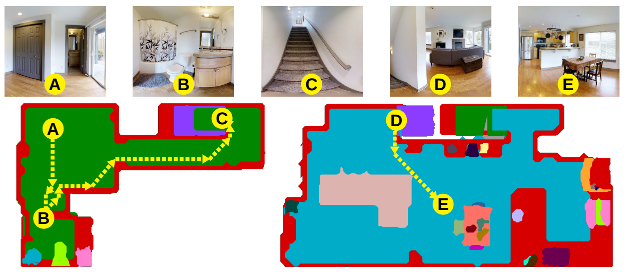


Fig. 6: Optimal path for the *Benevolence* mission shown in Table II.

we used 5 initial positions across different floors and rooms.

We use GPT-4 [51] to translate missions to LTL formulas, and Spot [52] for LTL formulas to automata as described in Sec. III. Following Sec. IV-D, we use GPT-4 [51] to generate the LLM heuristic function h_{LLM}. Given a scene graph $\mathcal{G}$, the mission described by the automaton $\mathcal{M}_\phi$, the LTL heuristic h_{LTL}, and the LLM heuristic h_{LLM}, we construct the hierarchical planning domain and run AMRA*.

Fig. 6 shows a path in *Benevolence* for the mission in Table II. Starting from the empty room on floor 0, the agent goes to the bathroom entrance without approaching the sink, and then proceeds upstairs to floor 1, finally reaching a chair in the dining room without entering the living room.

We compare different setups to investigate the effect of each hierarchy levels and the benefit of using our LLM heuristic. With all hierarchy levels having an LTL heuristic, we design 7 setups: occupancy level only without LLM heuristic (A*), all levels available but without LLM heuristics (NO-LLM), all levels with LLM heuristics (ALL), and one of the levels with LLM heuristic (OCC, OBJ, ROOM, FLR). Fig. 7 shows that the ALL setup manages to return a feasible path much faster than others thanks to the LLM guidance across all hierarchical levels, while it also approaches the optimal solution within similar time spans.

As an anytime algorithm, AMRA* starts searching with large weights on the heuristics, then reuses the results with smaller heuristic weights. As the planning iterations increase, the path gets improved. When the heuristic weight decays to 1, we obtain an optimal path. To further investigate the benefits of using LLM heuristics, we compare the number of node expansions per planning iteration. As shown in Fig. 8, applying our LLM heuristic to any hierarchy level reduces the node expansions significantly, which indicates that our LLM heuristic produces insightful guidance to accelerate AMRA*. An exciting observation is that the more hierarchy levels we use our LLM heuristic in, the more efficient the algorithm is. This encourages future research to exploit scene semantic information further to accelerate planning.

AMRA* allows a robot to start executing the first feasible path, while the path optimization proceeds in the background.

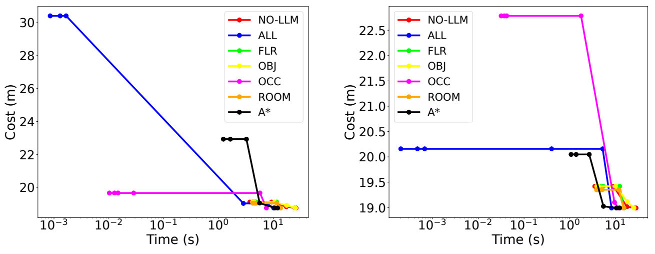


Fig. 7: Path cost vs planning time for different AMRA* variants.

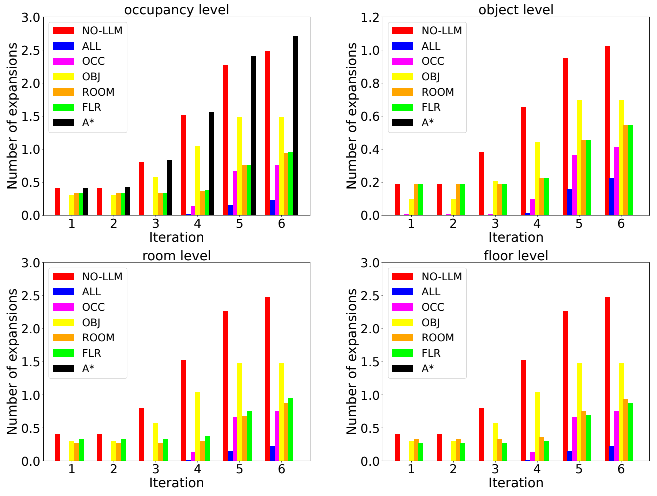


Fig. 8: Number of node expansions v.s. the planning iteration. Each sub-figure presents a hierarchy level.

TABLE III: First feasible path computation time, relative cost between first and optimal path, and optimal path computation time averaged over 5 initial conditions for mission 1 in *Benevolence*.

	1st iter. time (sec.)	1st iter. $cost/cost_{opt}$	opt. time (sec.)
ALL	**0.0007**	1.3763	8.9062
OCC	0.0244	1.0827	**8.3144**
OBJ	3.7460	1.0387	24.936
ROOM	3.6878	**1.0306**	13.1352
FLR	3.8106	1.0369	13.3287
NO-LLM	3.3260	1.0318	24.2516
A*	1.1202	1.0997	11.7594

TABLE IV: First path computation time, relative cost between first and optimal path, and optimal path computation time averaged over each scene's 5 missions and 5 initial positions/mission.

	Allensville(1-floor)	*Benevolence*(3-floor)	*Collierville*(3-floor)
ALL	0.24/1.69/3.50	**0.32**/1.24/**11.82**	**0.009**/1.26/**5.48**
NO-LLM	0.36/1.34/6.33	2.82/1.18/21.57	1.16/1.38/8.13
A*	**0.15/1.08/3.23**	1.33/**1.16**/14.22	1.19/**1.06**/7.04

Tables III and IV shows the time required to compute the first path, the path cost relative to the optimal path, and the time required to find an optimal path. The ALL configuration outperforms other setups in speed of finding the first path and the optimal path when the scene gets more complicated.

VI. CONCLUSION

We demonstrated that an LLM can provide symbolic grounding, LTL translation, and semantic guidance from natural language missions in scene graphs. This information allowed us to construct a hierarchical planning domain and achieve efficient planning with LLM heuristic guidance. We managed to ensure optimality via multi-heuristic planning, including a consistent LTL heuristic. Our experiments show that the LLM guidance is useful at all levels of the hierarchy for accelerating feasible path generation.

References

[1] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[2] S. Bowman, N. Atanasov, K. Daniilidis, and G. Pappas, "Probabilistic Data Association for Semantic SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[3] J. Dong, X. Fei, and S. Soatto, "Visual-inertial-semantic scene representation for 3d object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3567–3577.

[4] M. Shan, Q. Feng, and N. Atanasov, "OrcVIO: Object residual constrained Visual-Inertial Odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[5] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs," *The International Journal of Robotics Research (IJRR)*, vol. 40, no. 12-14, pp. 1510–1546, 2021.

[6] E. Zobeidi, A. Koppel, and N. Atanasov, "Dense Incremental Metric-Semantic Mapping for Multi-Agent Systems via Sparse Gaussian Process Regression," *IEEE Transactions on Robotics (T-RO)*, vol. 38, no. 5, pp. 3133–3153, 2022.

[7] A. Asgharivaskasi and N. Atanasov, "Semantic OcTree Mapping and Shannon Mutual Information Computation for Robot Exploration," *IEEE Transactions on Robotics (T-RO)*, vol. 39, no. 3, pp. 1910–1928, 2023.

[8] J. Wilson, Y. Fu, A. Zhang, J. Song, A. Capodieci, P. Jayakumar, K. Barton, and M. Ghaffari, "Convolutional Bayesian Kernel Inference for 3D Semantic Mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8364–8370.

[9] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison, "SceneCode: Monocular Dense Semantic Reconstruction Using Learned Encoded Scene Representations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] C. Estrada, J. Neira, and J. Tardos, "Hierarchical SLAM: real-time accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[11] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera," in *IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 5664–5673.

[12] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans," in *Robotics: Science and Systems (RSS)*, 2020.

[13] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *Robotics: Science and Systems (RSS)*, 2022.

[14] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber, "Semantic information for robot navigation: A survey," *Applied Sciences*, vol. 10, no. 2, 2020.

[15] 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*, vol. 32, no. 4, pp. 64–76, 2011.

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *Advances in neural information processing systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint:1810.04805*, 2018.

[18] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and Efficient Foundation Language Models," *arXiv preprint:2302.13971*, 2023.

[19] W. Chen, S. Hu, R. Talak, and L. Carlone, "Extracting zero-shot common sense from large language models for robot 3D scene understanding," *arXiv preprint:2206.04585*, 2022.

[20] ——, "Leveraging large language models for robot 3D scene understanding," *arXiv preprint arXiv:2209.05629*, 2022.

[21] D. Shah, B. Osiński, B. Ichter, and S. Levine, "LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on Robot Learning (CoRL)*, 2022, pp. 492–504.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, 2021, pp. 8748–8763.

[23] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[24] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-Logic-Based Reactive Mission and Motion Planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[25] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers," *arXiv preprint:2306.06531*, 2023.

[26] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, 2004, pp. 152–166.

[27] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2011, pp. 313–314.

[28] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 4817–4822.

[29] J. Fu, N. Atanasov, U. Topcu, and G. J. Pappas, "Optimal temporal logic planning in probabilistic semantic maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3690–3697.

[30] P. Purohit and I. Saha, "DT*: Temporal logic path planning in a dynamic environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3627–3634.

[31] Y. Kantaros and M. M. Zavlanos, "STyLuS*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research (IJRR)*, vol. 39, no. 7, pp. 812–836, 2020.

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

[33] J. Kim, A. Desrochers, and A. Sanderson, "Task planning and project management using petri nets," in *IEEE International Symposium on Assembly and Task Planning*, 1995, pp. 265–271.

[34] P. Lv, G. Luo, Z. Ma, S. Li, and X. Yin, "Optimal multi-robot path planning for cyclic tasks using petri nets," *Control Engineering Practice*, vol. 138, p. 105600, 2023.

[35] S. Karaman, S. Rasmussen, D. Kingston, and E. Frazzoli, "Specification and planning of uav missions: a process algebra approach," in *American Control Conference (ACC)*, 2009, pp. 1442–1447.

[36] Y. Oh, R. Patel, T. Nguyen, B. Huang, M. Berg, E. Pavlick, and S. Tellex, "Hierarchical planning with state abstractions for temporal task specifications," *Autonomous Robots*, vol. 46, no. 6, pp. 667–683, 2022.

[37] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal methods in system design*, vol. 19, pp. 291–314, 2001.

[38] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-World Perception for Embodied Agents," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] D. M. Saxena, T. Kusnur, and M. Likhachev, "AMRA*: Anytime multi-resolution multi-heuristic A*," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3371–3377.

[40] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[41] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 1-3, pp. 224–243, 2016.

[42] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 2997–3004.

[43] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, “Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search,” *The International Journal of Robotics Research (IJRR)*, vol. 39, no. 5, pp. 543–567, 2020.

[44] D. Shah, M. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine, “Navigation with large language models: Semantic guesswork as a heuristic for planning,” in *Conference on Robot Learning (CoRL)*, 2023.

[45] I. Kostavelis, K. Charalampous, A. Gasteratos, and J. K. Tsotsos, “Robot navigation via spatial and temporal coherent semantic maps,” *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 173–187, 2016.

[46] S. Amiri, K. Chandan, and S. Zhang, “Reasoning with scene graphs for robot planning under partial observability,” *IEEE Robotics and Automation Letters (RAL)*, vol. 7, no. 2, pp. 5560–5567, 2022.

[47] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone, “Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9272–9279.

[48] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *arXiv preprint arXiv:2209.07753*, 2022.

[49] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, “Task and motion planning with large language models for object rearrangement,” *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2086–2092, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257496672

[50] A. Pnueli, “The temporal semantics of concurrent programs,” *Theoretical computer science*, vol. 13, no. 1, pp. 45–60, 1981.

[51] OpenAI, “GPT-4 Technical Report,” in *arXiv: 2303.08774*, 2023.

[52] A. Duret-Lutz, E. Renault, M. Colange, F. Renkin, A. G. Aisse, P. Schlehuber-Caissier, T. Medioni, A. Martin, J. Dubois, C. Gillard, and H. Lauko, “From Spot 2.0 to Spot 2.10: What’s new?” in *International Conference on Computer Aided Verification (CAV)*, 2022, pp. 174–187.

[53] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, “ChatGPT for robotics: Design principles and model abilities,” Microsoft, Tech. Rep. MSR-TR-2023-8, February 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/chatgpt-for-robotics-design-principles-and-model-abilities/