Construal Level Theory for Agent-based Planning

Christopher McClurg[1,*], Alan R. Wagner[1] and Sarah Rajtmajer[2]

[1] *Department of Aerospace Engineering, The Pennsylvania State University, State College, PA, 16801*
[2] *College of Information Sciences and Technology, The Pennsylvania State University, State College, PA, 16801*

Abstract
Construal level theory (CLT) suggests that a person creates abstract mental representations, known as construals, in order to generate predictions, form counterfactuals, and guide behavior with respect to distal times, places, and actions [1]. This paper takes a step towards implementing CLT in agent reasoning; the impact of abstraction level on an ability to scavenge for needed items and ingredients is investigated. Our approach was parametrically tested in a Minecraft environment. Results show that planning with construals increased trial success rate by 14.8% as compared to planning without construals. Our work lays the foundation for a family of cognitively-plausible models that would allow computational agents to generate predictions about future events and valuations of future plans based on very limited prior training.

Keywords
AI, robotics, continual planning, Construal Level Theory, abstraction

1. Introduction

Construal level theory (CLT) suggests that a person creates abstract mental representations, known as contruals, in order to generate predictions, form counterfactuals, and guide behavior with respect to distal times, places, and actions [1, 2, 3]. In this process, an individual uses prototypical experiences from memory to generate a rough estimate of some future situation. As psychological distance - the subjective experience that something is close or far away - to a future event decreases, the prototypical experience moves from rough estimate to actionable plan by decorating the mental construct with additional information. The person uses this evolving mental construct to generate increasingly specific counterfactual alternatives to aid in decision making. Experimental evidence supporting the concepts underlying CLT have demonstrated the use of a cognitive construal process for visual perception, categorization, and action identification [3, 4, 5, 6, 7].

A simple example can demonstrate how planning is steered by a construal process, as described by CLT. In agreeing to provide a "birthday cake" for a party, an individual will initially recall prior celebrations to produce rough expectations regarding the actions needed to provide a birthday cake. At this stage, the plan to obtain a birthday cake is construed abstractly containing little or no actionable details. As the time to the party approaches, the person begins to mentalize

AAAI 2022 FALL SYMPOSIUM SERIES, Thinking Fast and Slow and Other Cognitive Theories in AI, November 17-19, Westin Arlington Gateway in Arlington, Virginia, USA
[*]Corresponding author.
cam.7498@psu.edu (C. McClurg); alan.r.wagner@psu.edu (A. R. Wagner); smr48@psu.edu (S. Rajtmajer)

more specific details about the task: the number of invited guests, food allergies or preferences of the celebrant, distance to local stores, etc. Expectations evolve as the mental construal of the plan for “obtaining a birthday cake” is decorated with increasing detail; the person eventually commits to a specific plan of action to obtain a cake. At the party, the person receives low-level feedback: number of guests present versus expected, reviews of the cake type, details of a new bakery, etc. As time passes, rather than store every detail of feedback, the person forms increasingly abstract construals, such as stereotyped preferences.

Our overarching research goal is to develop agents that use a CLT-like process to generate construals about future events and use these construals to make efficient, robust, general purpose plans of action that evolve as the situation evolves. We believe that the use of CLT could one day allow an agent to generate predictions about future events and valuations of future plans based on very limited prior training. This stands in contrast to traditional reinforcement learning paradigms that demand extensive training and are often limited by prior or evolving knowledge of a task’s reward structure. Ultimately we hope that our work will contribute the development of RL agents that operate across abstract representations and time periods, thus allowing an RL agent to plan both for the future and for the present.

This research represents the initial step toward a CLT process for an artificial agent or robot. Specifically, we evaluate different methods for creating construals and using these construals to develop a plan. The primary contributions of this paper is the development and evaluation of different construal processes that vary in their approach to creating abstract representations. We evaluate these processes in a scavenger hunt style experimental paradigm implemented in Minecraft. We chose this paradigm because hunting for objects is a temporal planning process involving objects and locations that can be represented at various levels of abstraction. Moreover, scavenger hunts can be related to simplistic variations of a variety of different logistics problems. Conceptually, the agent’s task is, given a list of items, identify where to search to collect the items on the list.

In our experiments, an agent is placed on an unfamiliar map, where only labels specifying types of locations (e.g. farm, house, etc.) and the agent’s position on the map are known. The agent’s task is to obtain the ingredients needed to make an item. The agent must find the ingredient, or find the ingredients necessary to make the ingredient, etc. Our construal process allows the agent to generate predictions about where an item or ingredient is placed in the world from exogenous general-purpose knowledge provided by ConceptNet [8] and to then form alternate plans of action to obtain the item or ingredient. ConceptNet is a semantic network that captures and relates the meanings of a large variety of common words [8]. We assume that items within a given distance of one another in the network form a superclass of items. Along with environmental observations by the agent, item similarity can be used to create more abstract construals that allow for more flexible planning.

In the sections that follow, we review related work on planning with a priori knowledge. Sections III and IV describe the details of the computational model and experimental setup. Section V discusses the results of the test. Finally, Section VI gives conclusions and future directions for this work.

2. Related Work

To the best of our knowledge, CLT has not previously been used for autonomous planning or learning. As such, much of the related work focuses on other approaches to creating and using abstractions, along with the different types of abstractions which have been used. For example, there has been a significant body of work attempting to incorporate ad-hoc a priori knowledge in the planning process. This knowledge typically has been formalized as semantic networks or description logic. Researchers have also investigated methods for abstracting meaning from robot experience, which is referred to as semantic mapping. These areas of research are described below: semantic networks, description logic, and semantic maps.

Semantic networks, also known as knowledge graphs (KG), are a form of knowledge representation often incorporated into task-planning. Galindo et. al hand-coded a semantic network to incorporate spatial and conceptual relations of items [9]. This KG informed navigation and planning, showing that a robot could infer the unknown location of an item. Ahn and Smith attempted to use the ConceptNet KG to break down a high-level, user-given task into sub-tasks [10]. Ultimately, the authors decided to hand-encode a domain-specific KG related to human daily-life actions. Xiong et. al used the ConceptNet KG to reason through uncertainty in task-planning [11]. The proposed algorithm groups KG relation types into high-order, parallelized agents that can form analogies to relate unknown situations to what is known. This algorithm also uses perceptual feedback to inform the working KG. Boteanu et. al proposed an algorithm that used ConceptNet to repair a plan [12]. Specifically, a robot would attempt a user-given task and if it required an item that was missing, then the robot would consult the knowledge graph to find a viable substitute. Lu et. al proposed an algorithm using the KGs FrameNet and WordNet. In this algorithm, the robot uses the KG to reason about verbs in user-given tasks, then converts the results into Action Set Programming (ASP) rules, then used the result to give the robot low-level actions [13]. Wang et. al propose a method for creating a domain-specific knowledge graph related to combat UAVs, but the method requires manual additions to build the KG [14]. More recent work has focused on creating KG-grounded reinforcement learning, such as that proposed by Xu et. al [15]. This work examines chat bot conversation planning while interacting with humans.

Description Logic (DL) is another form of knowledge representation incorporated into task-planning. KnowRob is a system that incorporates DL from perception, human observations, websites, and web-based knowledge bases [16, 17]. One motivation of this work was to address the symbol grounding problem, which is the “link between the abstract knowledge representation and a particular control system of a robot” [16]. The KnowRob system has been used in many applications. One example is a robot learning the organizational structure of a home in order to pick or place an item [18]. In this study, the KnowRob system is extended to include manufacturer-specific knowledge about products in the kitchen.

Semantic mapping is a process for creating meaning from robot experience, extracting relationships between semantic and spatial definitions. Kollar et. al showed that a robot can learn spatial- semantic knowledge from task-based human-robot dialogue [19]. In this study, a robot parses natural language and probabilistically maps the parsed entities to building locations. Liu and Wichert proposed a method for abstraction and inference to learn spatial-semantic knowledge of indoor environments [20]. This method takes user-defined abstract terms (e.g., type, relation) and task-specific knowledge to form a Markov Logic Network (MLN), where training data of 2D maps are used to form probabilistic maps of the environment. There is a great deal of research considering the general area of semantic mapping; for more information, see a survey by Kostavelis and Gasteratos [21]. While semantic mapping seeks to extract high-level meaning from experience, previous work does not consider a temporally focused framework such as CLT in which the level of abstraction increases with increasing past temporal distance.

3. Process Description

This paper proposes a process for using construals to generate agent plans. A flow chart for the process is shown in Figure 1. The main components of the process are: the input; the construal generation process; action generation; feedback; and output. These are discussed further below.

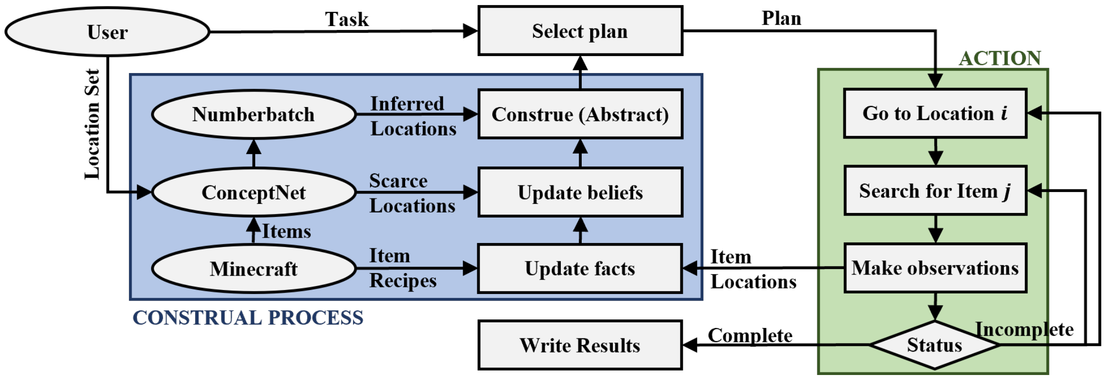

Figure 1: A process for planning with CLT. The main components are inputs (ovals), construal process (blue), action (green), and feedback (arrow annotated with item locations).

We consider as an example an agent seeking items for a recipe, a type of scavenger hunt task. The information needed to find these ingredients must come from somewhere. For humans, this information originates with experience in the real world. For an artificial agent or robot, it might be best if the agent or robot could retrieve information about items of interest from some available source of knowledge. We use ConceptNet as a source of related information for the agent. ConceptNet is a semantic network that relates the meanings of words and is comprised of over 34-million relations [8]. ConceptNet includes a variety of information about the items that the agent is seeking, possibly, but not always, including potential locations where the item might be found. Importantly, although ConceptNet provides a great deal of background information, much of the information lacks context and is difficult to marshal for solving a task. For example, if one searches ConceptNet for “bread” one listed location for “bread” is “the wallet of a high earner” which may not be relevant.

We use ConceptNet both directly and indirectly to create high-level (more abstract) construals for planning. Direct use of ConceptNet includes reading the knowledge graph and counting the number of relations needed to reach one item from another. A key assumption of this strategy is that items related to one another are members of a class. For example, “butter” and “cake” are related to "bread" and therefore in the same class. We can further aggregate classes to superclasses (higher levels of abstraction).

Indirectly, we use ConceptNet to generate a semantic similarity score for items of interest. We use ConceptNet Numberbatch to generate a set of pre-computed semantic vectors that give a snapshot of word meaning [8]. As before, we assume that items within a given similarity distance (cosine) are members of the same class and that classes can be aggregated into superclasses for varying level of abstraction. Returning to our example of "bread", we note the distances between the following pairs of items: "butter–cake" (0.648), "bread–cake" (0.550), and "butter–bread" (0.498). At a low level of construal (most concrete), only the most similar items are within the same class; therefore, the similarity distance threshold is small, say 0.5. Only "butter" and "bread" would be within the same class. However, this information may not be sufficient to find "bread", especially if the location of "butter" is unknown. So we consider a superclass of items by extending the similarity distance threshold, say to 0.8. Now, all of these items are within the same superclass. If any one of the items has a known location, then this information may be used to infer the location of "bread."

3.1. Inputs

Inputs to the process are denoted by ovals in Figure 1. The user can specify parameters such as the version of Minecraft, map complexity, task, and penalty weights. The version of Minecraft used in this experiment was V.1.11.2, which is compatible with the Malmo platform. Map complexity determines the location set, which is the set of unique labels for the custom map. Map complexity is discussed further in the next section. A task for this experiment requires the agent to obtain or create a given item, which is randomly selected from the item set in Minecraft. Finally, penalty weights in equation (2) are used to rank the available plans by their attributes: number of locations within the plan and level of abstraction required to make the plan.

The other inputs are Minecraft, ConceptNet and Numberbatch. The specific Minecraft version determines the set of items available in an environment and the corresponding recipe set. ConceptNet provides background knowledge in the form of concepts to the system. Each concept in ConceptNet may have many relations to other concepts. In this work, only relations specifying "at location" and "related to" were considered. Therefore, for the item set, a corresponding set of locations and related items could be obtained from ConceptNet. Numberbatch provides the agent with a numerical method to reason over the similarity of ConceptNet concepts.

3.2. Construal Process

The construal process is highlighted in blue in Figure 1. Recall that the purpose of the construal process is to to generate estimates and predictions about future events based on very limited prior training. These estimates and predictions are meant to guide the development of robust plans of action. The construal process begins with the concrete "facts" of the environment, defined by the agent's observations of the environment and the game's own constraints. Next in the construal process, "beliefs" act as the lowest-level (least abstract/most concrete) construal, providing only a general prediction for the location of items in the environment. These general predictions of item locations comes from ConceptNet, where game items are given as input and

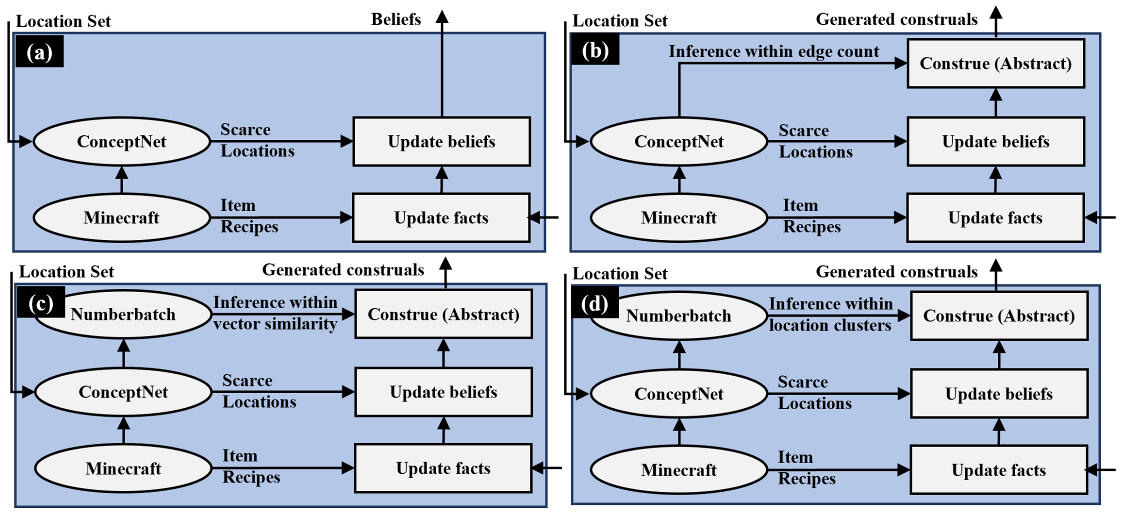

Figure 2: A comparison of the construal type: (a) None, (b) Direct Retrieval, (c) Similarity, (d) Clustering.

the output from ConceptNet is matched to location labels on the map.

We use the term *trial* to signify a new round of scavenging for recipe items by the agent. At the start of a trial, the agent has yet to make observations. The agent's belief consists of only items in the recipes and scarce predictions about where these items are located. These initial scarce predictions were only included in order to give the agent some initial direction other than random. As the trial continues, agent observations update the "facts" and any false predictions in "belief" are also corrected.

The agent uses ConceptNet and/or Numberbatch to infer the location of items. These inferences result from reasoning of the form: Because X1 is "located at" Y and X2 is "similar to" X1, then X2 is "located at" Y with some level of uncertainty. These inferences serve as higher-level (more abstract) construals. Uncertainty increases as abstraction is increased and "similar to" becomes more relaxed. The level of uncertainty created from this construal process is accounted for when ranking the available plans, as determined by equation 1. Note that the plan penalties (P_L and P_A) are discussed in Section IV.

$$S = 1 - P_L P_A \tag{1}$$

This plan ranking becomes especially useful when comparing one plan which has a longer route but is concrete (created without abstraction) to a plan with a shorter route that is based on an abstraction. Because the former plan exists without making predictions, there is little-to-no uncertainty and this may be the better option. The penalties due to distance and uncertainty are temporally motivated. We assume that the agent is trying to complete the plan promptly and thus, would ideally minimize both required abstraction and distance needed. However, abstraction does allow for planning at varying temporal distances, such that the penalties would be weighted less if the agent is not required to urgently complete the task. We use three discrete levels of abstraction to make inferences about the locations of items in the environment. Abstractions are created from the "facts," or instantaneous observations from the agent.

This study aims to evaluate both methods for creating construals and mechanisms by which construals can be used to inform a planning process. Figure 2 compares different construal process types: None, Direct Retrieval, Similarity, and Clustering.

1. No construal process. This model will act as a baseline in this study. As with all other cases, beliefs are initialized with the recipe set and sparse item locations. However, there are no inferred locations. The agent relies entirely on environment observations to update its beliefs and plans.
2. Direct Retrieval. In this method, inferred locations come from directly searching ConceptNet. At the lowest level of abstraction, item locations are inferred from items within one "related to" jump. For example, since X1 is "related to" X2 "located at" Y, X1 is "located at" Y with uncertainty. The next level of abstraction would allow for two "related to" jumps, etc.
3. Similarity. In this method, inferred locations come from using Numberbatch to obtain item vectors. Cosine similarity is then used to reason about similarity between any two items. At a low level of abstraction, locations are inferred from items within a small cosine distance. As abstraction is increased, the allowable cosine distance for two items to be considered similar is increased. An example is that the cosine distance between "door" and "gate" is 0.422, while the cosine distance between "door" and "stairs" is 0.734. A low level of abstraction may allow inference for the location of "door" from "gate", while a higher level of abstraction would be needed to infer location from "stairs."
4. Clustering. In this method, inferred locations come from using Numberbatch to obtain item vectors. Location centroids are then calculated from the item vectors at a given location. This process is referred to as agglomerative clustering: vectors within a specified distance of one another are averaged. A natural consequence of this process is that a given location could have multiple centroids, which allows agglomerative clustering to handle cases of high intra-class variance. At the lowest level of abstraction, item locations are inferred from centroid(s) within a small cosine distance and centroids within this small distance are merged with one another. As abstraction is increased, the allowable cosine distance for an item and centroid to be considered similar is increased.

As an example, consider the situation where a robot has already discovered "door" and "stairs" at location X. Note that the cosine distance between the vectors of "door" and "stairs" is 0.734. Now, consider when the robot decides the item "gate" may be beneficial to accomplishing the current scavenger task. The robot then tries to use a construal process to predict the location of the "gate." At a low construal level, the distance threshold is limited to a low value (say 0.5). The *similarity* method would use the cosine distance from "gate" to "door" (0.422) to infer that location X could contain "gate." The *clustering* method, on the other hand, would first generalize the items within a location, forming clusters of items within distance threshold. When the distance threshold is low (0.5), "door" and "stairs" (0.734) would not be clustered together, rather they would represent their own clusters. For a higher construal level (distance threshold 0.8), these items would form a single cluster and "gate" would be compared to the centroid of this new cluster.

3.3. Action

Action processes are highlighted in green in Figure 1. The agent is given a plan, which contains a count of locations and a count of items at each of these locations. The agent iterates through the plan, going to each location, and searching for specific items at that location. The agent uses the outcome of the search to assess the current situation. If the most recent search is successful and the current plan is complete, the trial is complete. Results are written to file. If the most recent search is successful but the current plan is not complete, then the agent continues the current plan. If the most recent search is unsuccessful, then the current plan is over. The agent determines whether the limit of plans attempted (20) has been exceeded. Until this is the case, the agent can re-plan using observations. When the agent has reached the limit, the trial is complete.

When the agent completes a trial, results are written to file. The results give the statistics for each trial: cumulative distance, run time, plan time, count of plans attempted, and success status. These statistics provide the basis for analysis.

3.4. Feedback: Item Locations

The feedback for the construal process comes from observations, as shown in Figure 1. Positive observations are the learned locations of items for which the agent was not directly looking. Negative observations include locations at which the search for an item fails. Both observations are used to update the "facts" about the environment.

4. Experimental Setup

The purpose of this experiment was to compare and characterize different construal creation methods. These methods were tested on a scavenger hunt task in the Minecraft simulation environment [22]. Controlled parameters are shown in Table 1, several of which are explained in greater detail below. Ten random trials were completed for each unique set of parameters, resulting in a total count of 1,200 trials.

Table 1
A summary of the controlled parameters.

Variable	Options	Count
Trial start[1]	*random*	1
Task item[1]	*random*	10
Plan scoring[2]	*fixed*	1
Item distribution[3]	*fixed*	1
Map complexity[4]	low, medium, high	3
Construal type	none, direct retrieval, similarity, clustering	4
Path planning[5]	A*	1
Trials/ condition	–	10

1. Randomness. For the random parameters, a seed was used so that this test could be repeated. For consistency, the trial count for each set of conditions was used as the seed.
2. Plan scoring. Equation (1) gives the scoring function for a given plan. The plan score was reduced by penalties for the count of required locations and required level of abstraction, given by equation (2). Penalty weights (W_L and W_A) were fixed at 1 in this study, but different weights may be investigated in future work.

$$P_L = W_L\left(\frac{n_{loc}}{n_{total}}\right) \quad P_A = W_A\left(\frac{n_{level}}{n_{max}}\right) \tag{2}$$

   A penalty was assigned based on the count of locations traveled. The more locations traveled to increases the penalty (P_L) because of the increased distance that the agent has to travel. A higher level of abstraction increases the penalty (P_A) because inferred locations are less certain than those actually observed by the agent.
3. Item distribution. In order to produce a distribution of items which was created independently of the researchers yet realistic, 108 human subjects were asked to complete a survey indicating where the items should be placed. The survey asked each respondent to select up to three location labels for approximately thirty unique items. The survey was conducted over Amazon Mechanical Turk, and each worker was paid \$2.50. This use of human subjects was IRB approved. Attention checks were used to verify that the workers were attentive. With the results from the survey, the items were placed at the most highly-rated location for the given map complexity (low, medium, or high), as specified by the trial conditions. The Mechanical Turk survey results are shown in Figure 3. Each plot reflects a unique item distribution in maps with low, medium and high complexity. Map complexity corresponds to the number of locations – a subset of all locations – on the map. The location "house" contains a majority of the game items in each case. The "post office" label was not chosen for any item. Item placement was determined by the most common location response by survey participants, constrained to the locations within a given subset (6, 12, or 18 locations).

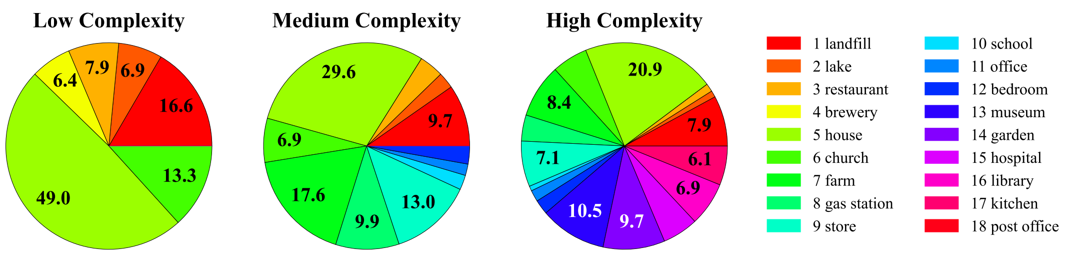

Figure 3: Item distribution (%) per map complexity, enabled by MTurk human survey.

4. Map complexity. The varying map complexity can be seen in Figure 4, where the boxes in the figure show the perimeter of locations. There are 18 potential locations, which are shown in the key. A map with low, medium, and high complexity contains 6, 12 and 18 of

these locations, respectively. The purpose of varying map complexity is to show how the planning process scales to a more complicated environment.

5. Agent navigation. The agent must move between locations as it follows a plan and within a location as it searches for the plan item. A coarse 50x50 maze was hand-created to represent the map. The A* algorithm was used to create a path between any two points, given the coarse maze as input. The algorithm produced a set of coordinates that the agent then followed. A position feedback control system was used to follow the path.

The test map (Figure 4) has been adapted from a map created by an architect on Planet Minecraft [23]. The scavenger hunt is defined by both recipes and items found in the game environment. The recipe set was generated by another contributor to Planet Minecraft [24]. The item set was obtained from DigMinecraft [25]. Lastly, the agent navigation within the game environment relies on the A* pathfinding algorithm [26].

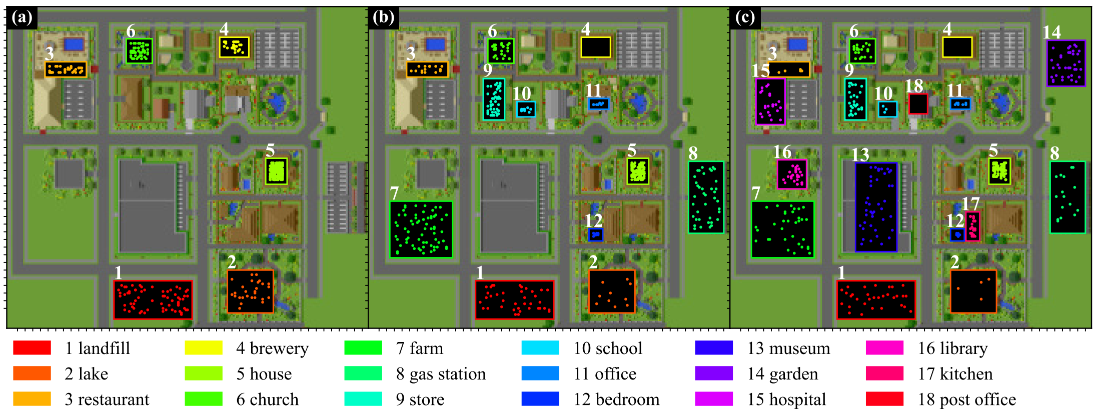

Figure 4: A comparison of map complexity: (a) Low, (b) Medium, and (c) High. Colored dots represent item placement.

5. Experimental Results

During the experiment, 1,200 trials were run. The process was implemented in Python 3.6 using multiprocessing (AMD Ryzen 9 5950X 16-core). The following trial statistics (dependent variables) were collected: run time, plan time, distance traveled, number of observations, number of attempts, and success status. Success/failure was determined by whether the agent was able to accomplish the task within 20 attempts. As shown in Table 1, the independent variables in this experiment were task item, map complexity, and the method for creating construals.

5.1. Task Item

Ten random task items were tested in this study, as shown in Table 2. With respect to the trial success rate, the agent performed best obtaining the items “jungle fence gate”, “boat”, and “sign.” The agent performed worst obtaining task items “spawn egg”, “tipped arrow”, and “lingering potion." This result is consistent with the fact that that ConceptNet consists of words, their meanings and relations. The most difficult to obtain task items are also the most game-specific, and thus involve the least commonly-used words in ConceptNet. In Table 2, there is not a clear trend between task item success rate and cumulative distance traveled in successful plans. There is also not a clear trend between task item success rate and survey agreement, calculated as the percentage of respondents that agreed in the best placement of given item.

Table 2
Results averaged over trials with respect to task item. Note that survey agreement is calculated as the percentage of respondents that agreed in the best placement of given item. *Indicates successful trials only.

Task Item	**Distance*** (10^5)	**Survey Agreement** (%)	**Success Rate** (%)
(0) diamond ore	23.0 ± 16.7	36.36	62.50
(1) stained glass	20.7 ± 15.9	40.00	82.50
(2) cake	15.3 ± 16.3	72.73	50.83
(3) jungle fence gate	**10.7 ± 10.3**	55.56	**92.50**
(4) boat	18.6 ± 14.1	40.00	89.17
(5) spawn egg	25.1 ± 19.8	50.00	41.67
(6) redstone ore	17.7 ± 15.6	44.44	79.17
(7) sign	16.4 ± 13.3	44.44	90.00
(8) tipped arrow	16.1 ± 14.7	38.46	44.44
(9) lingering potion	14.3 ± 13.1	40.00	39.66

5.2. Map Complexity

Three maps of varying complexity were tested in this study, each with a different number of location labels (6, 12, and 18). Table 3 shows results as map complexity is varied. As the number of locations is increased, success rate decreases. Also, as the number of locations is increased, the agent must travel further to successfully complete tasks.

Table 3
Results averaged over trials with respect to map complexity. *Indicates successful trials only.

Map Complexity	**Distance*** (10^5)	**Success Rate** (%)
(0) low	**12.3 ± 11.8**	**72.61**
(1) medium	19.9 ± 15.1	69.85
(2) high	21.2 ± 17.5	59.70

5.3. Construal Method

Three different methods for creating construals were tested in this study, as shown in Table 4. The construal methods showed a significantly increased trial success rate when compared to the case of no construal process (No vs. Di: (t(377)=2.7355, p=0.01), No vs. Si: (t(379)=2.0995, p=0.04), No vs. Cl: (t(376)=2.8204, p=0.01)). The success rates when comparing different methods for creating construals was not significantly different from each other (Di vs. Si: (t(424)=0.6510, p=0.52), Di vs. Cl: (t(421)=0.1145, p=0.91), Si vs. Cl: (t(423)=0.7591, p=0.45)).

Table 4
Results averaged over trials with respect to construal method. *Indicates successful trials only.

Construal Method	**Plan Time* (min.)**	**Distance* (10^5)**	**Success Rate (%)**
(0) none	**0.00 ±0.00**	21.3 ± 17.1	56.03
(1) direct retrieval	n/a	16.6 ± 14.3	70.93
(2) similarity	3.63 ± 2.96	17.0 ± 15.0	**71.83**
(3) clustering	1.45 ± 1.95	**16.1 ±14.7**	70.77

Additional factors should be considered when comparing construal methods. Table 4 gives the cumulative plan time and travelled distance in successful trials as construal process is varied. Note that the “direct retreival” method is not included in comparison, because it has significant pre-computational cost. Recall that the “direct retreival” method involves having the planner directly searching the ConceptNet knowledge graph; an API can be used for making requests and reading the knowledge graph, but this API is rate-limited. Thus, the highest performing construal method in terms of both cumulative plan time and distance is the ”clustering” construal process.

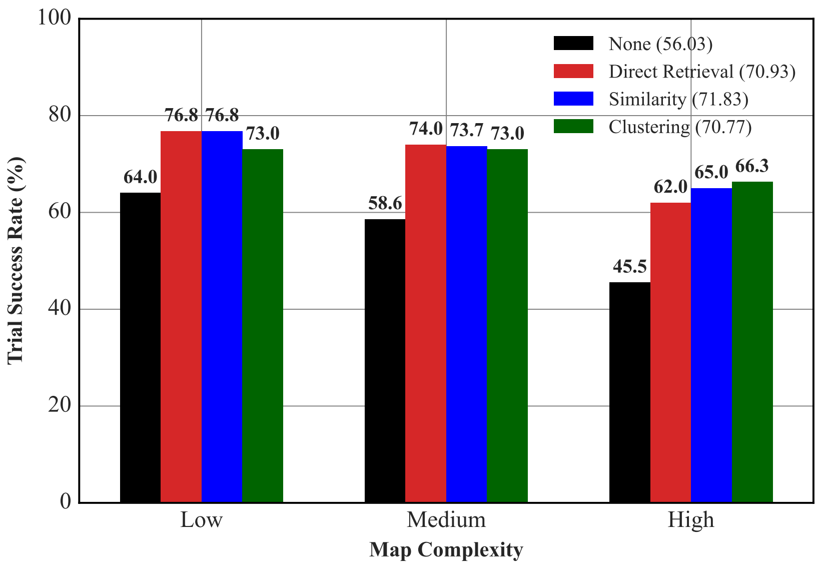

Figure 5: A plot of success rate (%) as a function of map complexity and construal type.

Figure 5 shows how each construal process construal process scales with respect to changing map complexity. The success rate of having no construal process depreciates most rapidly (−18.5%) from low to high complexity; this process relies entirely on observations to make a plan. The “direct retrieval” construal process is next in terms of depreciation (−14.8%), followed by the “similarity” construal process at (−11.8%). The best process in terms of depreciation is the “clustering” construal process (−6.7%).

6. Discussion

This paper proposes a method that connects a construal creation process to a system for planning. Our initial results of testing this process are encouraging. Success rates for simple tasks are significantly higher if construals are used. Planning with a construal process shows a 14.8% increase in trial success rate over all testing, as compared to *no* construal process (Table 4). Moreover, plan performance does not degrade as rapidly with complexity when using construals. Planning with a construal process shows lower depreciation (−6.7%) as the complexity of the map is increased, as compared to *no* construal process (−18.5%) (see Figure 5). Construals improve planning because the construal process gives the agent plan options that are not otherwise available via inferred locations.

The results of this study also provides a comparison of construal processes. The success rates of each construal process are relatively close (within 0.9%, see Table 4). The “direct retrieval” method is unattractive due to the relatively high pre-computational cost. The relative computational cost is lower for the “clustering” method (−60.0%), as compared to the “similarity” method. The “clustering” method showed the lowest depreciation rate (−6.7%) as the map complexity was increased, as compared to “direct retrieval” (−14.8%) and “similarity” (−11.8%) (see Figure 5). Moreover, the clustering method can be motivated and conceptually connected to recent methods for classification of items and locations [27, 28].

7. Conclusion

Construal Level Theory (CLT) suggests that people create mental construals to help guide their planning and decision-making [1, 2, 3]. We have attempted to create a computational process that notionally mimics construal level theory in that we afford an agent a means for representing items at various levels of abstraction during a scavenger hunt. Our experimental results indicate that creating and using construals increases the agent’s success rate and slows depreciation of performance in conditions of increasing complexity.

Given the increased importance that abstraction is playing within the machine learning community, this work provides foundational insight into how knowledge graphs can be used to generate abstractions over a wide range of data inputs. Our goal with this research was to demonstrate that a variety of different approaches could be used to create construals and that, regardless of the approach, the use of construals is beneficial. We believe that demonstrating that a construal process can originate from different underlying representations allows for more avenues for future research.

Acknowledgments

This work is supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0197.

References

[1] Y. Trope, N. Liberman, Temporal construal, Psychological Review 110 (2003) 403–421.

[2] Y. Trope, N. Liberman, C. Wakslak, Construal levels and psychological distance: Effects on representation, prediction, evaluation, and behavior, Journal of Consumer Psychology 17 (2007) 83–95.

[3] Y. Trope, N. Liberman, Construal-level theory of psychological distance, Psychological Review 117 (2010) 440–463.

[4] B. S. Moore, D. R. Sherrod, T. J. Liu, B. Underwood, The dispositional shift in attribution over time, Journal of Experimental Social Psychology 15 (1979) 553–569.

[5] N. Liberman, Y. Trope, The role of feasibility and desirability considerations in near and distant future decisions: A test of temporal construal theory, Journal of Personality and Social Psychology 75 (1998) 5–18.

[6] N. Liberman, M. D. Sagristano, Y. Trope, The effect of temporal distance on level of mental construal, Journal of Experimental Social Psychology 38 (2002) 523–534.

[7] K. Fujita, M. D. Henderson, J. Eng, Y. Trope, N. Liberman, Spatial distance and mental construal of social events, Psychological Science 17 (2006) 278–282.

[8] R. Speer, J. Chin, C. Havasi, Conceptnet 5.5: An open multilingual graph of general knowledge, in: Thirty-first AAAI conference on artificial intelligence, 2017.

[9] C. Galindo, J.-A. Fernández-Madrigal, J. González, A. Saffiotti, Using semantic information for improving efficiency of robot task planning, in: 2007 IEEE/International Conference on Robotics and Automation (ICRA), 2007.

[10] H.-i. Ahn, D. A. Smith, Action planning with commonsense knowledge, in: CHI'09 Extended Abstracts on Human Factors in Computing Systems, 2009, pp. 3565–3570.

[11] X. Xiong, Y. Hu, J. Zhang, Epistemebase: A semantic memory system for task planning under uncertainties, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 4503–4508.

[12] A. Boteanu, D. Kent, A. Mohseni-Kabir, C. Rich, S. Chernova, Towards robot adaptability in new situations, in: 2015 AAAI Fall Symposium Series, 2015.

[13] D. Lu, Y. Zhou, F. Wu, Z. Zhang, X. Chen, Integrating answer set programming with semantic dictionaries for robot task planning, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), Melbourne, Australia, 2017, pp. 4361–4367. doi:10.24963/ijcai.2017/609.

[14] S. Wang, Y. Zhang, Z. Liao, Building domain-specific knowledge graph for unmanned combat vehicle decision making under uncertainty, in: 2019 Chinese Automation Congress (CAC), 2019, pp. 4718–4721.

[15] J. Xu, H. Wang, Z. Niu, H. Wu, W. Che, Knowledge graph grounded goal planning for open-domain conversation generation, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 9338–9345. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6474. doi:10.1609/aaai.v34i05.6474.

[16] M. Tenorth, M. Beetz, Knowrob — knowledge processing for autonomous personal robots, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4261–4266. doi:10.1109/IROS.2009.5354602.

[17] M. Tenorth, M. Beetz, Knowrob: A knowledge processing infrastructure for cognition-

enabled robots, The International Journal of Robotics Research 32 (2013) 566–590.

[18] M. J. Schuster, D. Jain, M. Tenorth, M. Beetz, Learning organizational principles in human environments, in: 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 3867–3874. doi:10.1109/ICRA.2012.6224553.

[19] T. Kollar, V. Perera, D. Nardi, M. Veloso, Learning environmental knowledge from task-based human-robot dialog, in: 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 4304–4309. doi:10.1109/ICRA.2013.6631186.

[20] Z. Liu, W. Wang, D. Chen, G. von Wichert, A coherent semantic mapping system based on parametric environment abstraction and 3d object localization, in: 2013 European Conference on Mobile Robots, 2013, pp. 234–239. doi:10.1109/ECMR.2013.6698848.

[21] I. Kostavelis, A. Gasteratos, Semantic mapping for mobile robotics tasks: A survey, Robotics and Autonomous Systems 66 (2015) 86–103. URL: https://www.sciencedirect.com/science/article/pii/S0921889014003030. doi:https://doi.org/10.1016/j.robot.2014.12.006.

[22] M. Johnson, K. Hofmann, T. Hutton, D. Bignell, The malmo platform for artificial intelligence experimentation, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, AAAI Press, 2016, p. 4246–4247.

[23] P. M. niki2011, Minecraft maps/land structure/small-town, 2014. URL: https://www.planetminecraft.com/project/small-town-1896553/.

[24] P. M. leasoncre, Minecraft blogs/other/'the master list of items/mobs/blocks in minecraft', 2021. URL: https://www.planetminecraft.com/project/small-town-1896553/.

[25] DigMinecraft, Minecraft id list (java edition 1.11.2), 2021. URL: https://www.digminecraft.com/lists/item_id_list_pc_1_11.php.

[26] N. Swift, Nicholas-swift/astar.py, 2021. URL: https://gist.github.com/Nicholas-Swift/003e1932ef2804bebef2710527008f44.

[27] A. Ayub, A. R. Wagner, Cognitively-inspired model for incremental learning using a few examples, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 222–223.

[28] A. Ayub, A. R. Wagner, Centroid-based concept learning for RGB-D indoor scene classification, in: British Machine Vision Conference (BMVC), 2020.