GOALNET: Inferring Conjunctive Goal Predicates from Human Plan Demonstrations for Robot Instruction Following

Shreya Sharma[1] *, **Jigyasa Gupta**[1, 2*], **Shreshth Tuli**[3], **Rohan Paul**[1], **Mausam**[1]

[1]Indian Institute of Technology Delhi, India
[2]Samsung R&D Institute India, Delhi
[3]Imperial College London, UK

Abstract

Our goal is to enable a robot to learn how to sequence its actions to perform tasks specified as natural language instructions, given successful demonstrations from a human partner. The ability to plan high-level tasks can be factored as (i) inferring specific goal predicates that characterize the task implied by a language instruction for a given world state and (ii) synthesizing a feasible goal-reaching action-sequence with such predicates. For the former, we leverage a neural network prediction model, while utilizing a symbolic planner for the latter. We introduce a novel neuro-symbolic model, GOALNET, for contextual and task dependent inference of goal predicates from human demonstrations and linguistic task descriptions. GOALNET combines (i) *learning*, where dense representations are acquired for language instruction and the world state that enables generalization to novel settings and (ii) *planning*, where the *cause-effect* modeling by the symbolic planner eschews irrelevant predicates facilitating multi-stage decision making in large domains. GOALNET demonstrates a significant improvement (51%) in the task completion rate in comparison to a state-of-the-art rule-based approach on a benchmark data set displaying linguistic variations, particularly for multi-stage instructions.

1 Introduction

Robots are used in various scenarios where they interact with humans to perform tasks. Understanding natural language instructions, in such interactive settings, is crucial to effectively attain the human intended tasks. Recent efforts aim to directly map input state and language instructions to actions by inferring intended goals at the start to generate an action plan (Tuli et al. 2021; Bansal et al. 2020; Mei, Bansal, and Walter 2016; Suhr and Artzi 2018). Such methods tend to lack the ability to interact dynamically with humans or scale with environment complexity (Misra et al. 2018). Thus, we decompose the problem of generating an action sequence into the inference of task-specific goal predicates and subsequent generation of actions. However, learning to predict goal predicates is challenging due to following reasons. First, the hypothesis space of goal constraints may be large due to irrelevant or side-effects of other constraints. Second, high-level tasks may involve multi-stage plans requiring the learner to uncover the sub-goals characterizing the overall task from human demonstrations (Pflueger and Sukhatme 2015). Finally, the robot may encounter new instructions and world states necessitating conceptual generalization beyond limited training data.

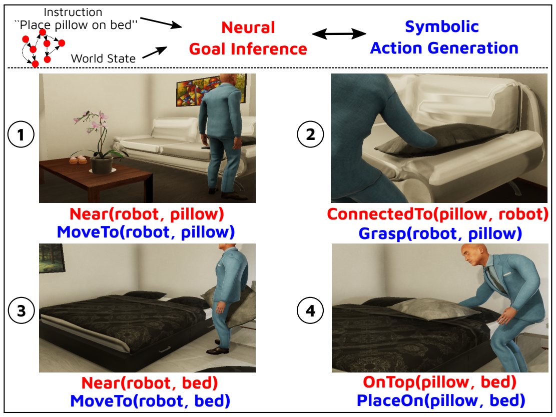


Figure 1: We consider the task of inferring conjunctive goal predicates (red) from input world state and language instruction such that when given to a symbolic planner (blue), we reach a goal state.

To make goal inference tractable in large-scale domains with multi-stage instructions, we leverage a symbolic planner with action models (in PDDL form) allowing us *explain-away* side-effect or irrelevant predicates. To handle linguistic variations, we utilize a crowd-sourced dataset to train a neural model to infer conjunctive goal predicates for an input world state and instruction. We conjecture that viewing multiple task demonstrations enable an agent to infer the predicates/constraints needed to reach the goal. We assume that the human demonstrates plans for varied natural language instructions (e.g., *"move all the fruits to the kitchen"*, *"fill the cup with water"* etc.). The human demonstration constitute language instructions with state-action transitions leading to goal-reaching states. Due to the high variability in human language, learning through demonstrations facilitates goal reaching even with unseen instruction inputs.

This paper addresses the problem of interpreting a task instruction from non-expert human users as a succinct set

[*]These authors contributed equally. Project page: https://github.com/reail-iitd/goalnet.

of goal predicates. For such users, we hypothesize that it is easier to demonstrate tasks in lieu of directly providing symbolic description. We propose a *neuro-symbolic model*, which we call GOALNET, to infer goal predicates/constraints for a given state of the environment and natural language instruction. These constraints can then be passed to a high-level *symbolic* planner to give out a sequence of actions in an interleaved fashion (see Fig. 1). This style of using a planner for forward simulation is unique to this work. GOALNET learns dense representations for instruction, world state and state history, enabling generalization to settings unseen during training. Our evaluation shows that the interactive dynamics between the neural model and the symbolic planner enables a robot to complete higher-level tasks such as cleaning a table, fetching fruits, arranging pillows, or even preparing meals in complex environments or scenes. Our results demonstrate a significant improvement of 51% in the task completion rate in comparison to a state-of-the-art heuristic and rule-based approach for a mobile manipulator in a kitchen or living-room like environment with multi-stage instructions and linguistic variations.

2 Related Work

Grounding Instructions to Symbols. A line of work (Thomason et al. 2015; Howard, Tellex, and Roy 2014) aims to train a supervised form of model that infers goal constraints for an input natural language instruction for classical instruction following. Similarly, Squire et al. (2015) ground language instructions to rewards functions. Such annotations are challenging to provide and time-consuming for the human operator. Instead, we adopt a framework in which only the natural demonstrations are provided to a learner having access to the full environment state in lieu of explicit knowledge of constraints corresponding to the input instructions.

Inferring plans from instructions. Alternatively, researchers have considered the problem of contextual induction of multi-step plans from an instruction. The seminal work of Misra et al. (2016) presents a probabilistic model that integrates spatial cues and object properties in conjunction with a symbolic planner to determine a grounded action sequence for an instruction. In other work (Misra et al. 2018), they attempt to increase the horizon of plan induction by attempting to decouple the visual goal reasoning with motion planning. In related work, She and Chai (2016) present an incremental approach to interpret verbs in the form of a "constraint set" hypothesis spaces that can be acquired through an explicit feature-based representation. We build on these works and make the following contributions. First, we forego the need for hand-crafted features and learn a dense representation for the environment and language enabling generalization. Second, we cast the problem of goal constraint inference as a contextual sequence prediction problem to generate larger sets of constraints. Alternatively, researchers cast the problem of instruction following as imitating tasks demonstrated by a human partner. Examples include, (Tuli et al. 2021; Liao et al. 2019; Bragg and Brunskill 2020; Shridhar et al. 2020). The efforts aim at learning a policy that allows the robot or an embodied AI agent to complete tasks explicitly specified by a human in the form of a symbolic goal. This work address the complementary problem of inferring goal constraints and delegate the task of policy learning or planning to works as above.

Goal Inference. A growing body of work addresses the task of recognizing goals by observing the actions of agents and inferring a likelihood of each possible goal-predicate among the possible ones at each world state (Meneguzzi and Pereira 2021). The earliest work of Lesh and Etzioni (1995) introduced a goal recognizer that observes human actions to prune inconsistent actions or goals from an input graph state. (Baker, Saxe, and Tenenbaum 2009) pose goal inference as probabilistic inverse planning and estimating a posterior distribution over sub-goals by observing human-generated plans. (Mann 2021) presents an approach to infer goals even with sup-optimal or failed plans. Analogously, (Dragan et al. 2015) present a robot motion planner that generates legible plans using human-robot collaboration. (Boularias et al. 2015) utilize inverse reinforcement learning to infer preference over predicates by using cost functions via human demonstrations. This paper addresses the related but different problem of inferring the specific hypothesis space that constitutes a goal for a task, taking a data-driven approach assuming the presence of *human-optimal* demonstrations of the task in multiple contexts. The problem of disambiguation or attaining the intended goal is delegated to a planner or a learned policy.

3 Problem Formulation

Robot and Environment Models. We consider a robot as a autonomous agent with the ability to freely move and manipulate multiple objects in natural confined domains such as a *kitchen* or *living room* like environments. We consider objects as symbolic entities that consist of (i) identifying tokens such as "apple", "stove" and "pillow", (ii) object states such as $\text{Open}/\text{Closed}$, On/Off, and (iii) properties such as isSurface, isContainer, isGraspable, etc. We consider object relations such as (i) *support*: for example an apple supported on a tray or a shelf, (ii) *containment*: for instance a pillow inside a box, carton or a cupboard, (iii) *near*: an object being in close proximity to another, and (iii) *grasped*: robot grasping a graspable object such as microwave door, fork, tap, etc. Let s denote the state of a domain/environment in which an agent is expected to perform a task. The world state s is a collection of symbolic objects including their identifiers, states and properties. The world state also includes object relations such as OnTop, Near, Inside and ConnectedTo. We denote the set of spatial-relation types between objects and the set of object state constraints by $\mathcal{S}$. Examples include $\text{OnTop}(\text{pillow}_0, \text{shelf}_0)$, $\text{ConnectedTo}(\text{fork}_0, \text{robot})$ and $\text{stateIsOpen}(\text{tap}_0)$. Let s_0 denote the initial state of the domain and $\mathcal{O}(.)$ denote a map from an input state s to a set of symbolic objects $\mathcal{O}(s)$ populating the world state s. Let $\mathcal{R}(.)$ denote a map from an input state s to the set of object relations $\mathcal{R}(s)$. For any given state s with relation set $\mathcal{R}(s)$, a relation $r \in \mathcal{R}(s)$ is denoted by $R(o^1, o^2)$ that represents a relation of type $R \in \mathcal{S}$ between objects $o^1 \in \mathcal{O}(s)$ and $o^2 \in \mathcal{O}(s)$ or as $R(o^1)$ in case of r being an object state constraint.

Action and Transition Models. We denote the set of all possible symbolic actions that the agent can perform on objects as A. For any given state s, an action $a \in A$ is represented in its abstract form as $I(o^1, o^2)$ where the interaction predicate $I \in \mathcal{I}$ affects the state or relation between o^1 and o^2. The interaction set $\mathcal{I}$ includes action predicates such as Grasp, MoveTo, stateOn, PlaceOn. Interaction effects are considered to be deterministic. For instance, a Grasp action establishes a *grasped* relation between the agent and a isGraspable target object. A stateOn action applies to an isTurnable object, such as a tap, to swap its state between On and Off. A $\mathrm{MoveTo}(\mathrm{robot}, \mathrm{couch}_0)$ action establishes a *near* relation between the two objects robot and couch_0. Similarly, a $\mathrm{PlaceOn}(\mathrm{pillow}_0, \mathrm{couch}_0)$ action establishes a *support* relation from pillow_0 and couch_0. Interactions are also associated with pre-conditions in the form of relations or properties. For instance, PlaceOn and PlaceIn actions are allowed only when the object has a *grasped* relation with the agent. Also, a Grasp action is permitted only when the target object has the isGraspbable property. For more details see Appendix A. In our formulation, we assume presence of a low-level motion planner and delegate the problem of navigation and dexterous object manipulation to other works such as by Fitzgerald, Goel, and Thomaz (2021); Gajewski et al. (2019); Lee et al. (2015). We denote the deterministic transition function by $\mathcal{T}(\cdot)$. Thus, we can generate the successor state $s_{t+1} \leftarrow \mathcal{T}(a_t, s_t)$ upon taking the action a_t in state s_t.

Tasks and Goals. Given an initial state of the environment s_0 and transition model $\mathcal{T}$, the robot needs to perform a task in the form of a *declarative* natural language instruction l. An instruction l is encoded in the form of a sequence $\{l_0, \ldots, l_z\}$ where each element is a token. Each instruction l corresponds to two sets Δ_l^+ and Δ_l^-, both being sets of symbolic goal constraints among world objects. The presence of Δ_l^+ constraints and the absence of Δ_l^- constraints in the final state characterizes *a* successful execution for the input instruction and are referred to as *positive* and *negative* constraints respectively. For example, for the input state s with a pillow on the shelf and another inside a cupboard, the *declarative* goal, $l =$ "put the shelf pillow on the couch" can be expressed as sets of constraints $\Delta_l^+ = \{\mathrm{OnTop}(\mathrm{pillow}_0, \mathrm{couch}_0)\}$ and $\Delta_l^- = \{\mathrm{OnTop}(\mathrm{pillow}_0, \mathrm{shelf}_0)\}$. To successfully execute an input instruction l from an initial state s_0, the agent must synthesize a plan as a sequence of actions $\{a_0, \ldots, a_T\}$ such that the final state $s_T = \mathcal{T}(\ldots \mathcal{T}(s_0, a_0) \ldots, a_T)$ consists of the goal constraints Δ_l^+ and removes the constraints Δ_l^-. Let $\mathcal{G}(s, l)$ denote the *goal check* function that determines if the intended goal l is achieved by a state s as

$$\mathcal{G}(s, l) = \mathbb{1}\big((\Delta_l^+ \subseteq R) \wedge (\Delta_l^- \cap R = \emptyset)\big), \tag{1}$$

where $R = \mathcal{R}(s)$ and Δ_l^+, Δ_l^- are constraint sets for instruction l. We represent the set of all states that give $\mathcal{G}(s, l) = 1$ as S^l and refer to them as goal states in further discussion. We denote the positive and negative constraints established at each step from state s_t to s_{t+1} by δ_t^+ and δ_t^-. We also denote the *constraint history* till time t by $\eta_t = \{(\delta_0^+, \delta_0^-), \ldots, (\delta_{t-1}^+, \delta_{t-1}^-)\}$.

Learning to Reach Goals. This work aims to reach a goal state, *i.e.* $\mathcal{G}(s_0, l) = 1$, given an initial world state s_0 and a natural language instruction l. We consider a discrete-time control problem of producing a goal-reaching plan by producing a policy that estimates an action sequence. Predicting actions for high-level multi-stage instructions can be partitioned into first predicting the required goal constraints to be achieved and then producing an action (Misra et al. 2018). We focus on learning *how* to infer goal-predicates and leverage a parameterized function $f_\theta(.)$ with parameters θ to determine the predicates to be achieved for the given state, constraint history and input instruction as

$$\delta_t^+, \delta_t^- = f_\theta(s_t, l, \eta_t). \tag{2}$$

We also leverage a given planner $\mathcal{P}(.)$ that takes state, goal predicates and symbolic cause-effect domain definitions (denoted by Λ) as inputs and generates an action sequence $\mathcal{P}(s_t, \delta_t^+, \delta_t^-, \Lambda)$. Thus, at time-step t, our action a_t is realized as $a_t = \mathcal{P}(s_t, f_\theta(s_t, l, \eta_t), \Lambda)$.

Thus, we perform an interleaved predicate inference (using f_θ) and feasible plan generation (using $\mathcal{P}$) at each time-step to reach a goal state. Formally, let $S_t^{\mathcal{P}, f_\theta, s}$ be a random variable denoting the state resulting from interleaved execution of planner $\mathcal{P}(.)$ and goal predictions $f_\theta(.)$ from state s for t time steps. We aim to learn $f_\theta(.)$ such that given to a planner $\mathcal{P}(.)$ the resultant state $S_k^{\mathcal{P}, f_\theta, s}$ reached in up to T steps *is a goal state* and the size of the set of inferred *predicate set is minimized*. Thus, we have

$$\begin{aligned}
\underset{\theta}{\text{minimize}} \quad & \|\delta_t^+ \cup \delta_t^-\| \\
\text{s. t.} \quad & \forall\, t, \delta_t^+, \delta_t^- = f_\theta(s_t, l, \eta_t), \\
& \forall\, t, \mathcal{P}(s_t, \delta_t^+, \delta_t^-, \Lambda) \text{ is executed.} \\
& \exists\, k \in \{1, \ldots, T\}, \text{ s.t. } \mathcal{G}(S_k^{\mathcal{P}, f_\theta, s_0}, l) = 1.
\end{aligned}$$

Our function $f_\theta(.)$ is trained as a likelihood prediction model for the goal constraints leveraging a crowd-sourced dataset of task demonstrations. We denote a dataset D_{Train} of N goal-reaching plans as

$$\mathcal{D}_{\mathrm{Train}} = \{(s_0^i, l^i, \{s_j^i, a_j^i\}) \mid i \in \{1, N\}, j \in \{0, t_i - 1\}\},$$

where the i^{th} datum consists of the initial state s_0^i, the instruction l^i and a state-action sequence $\{(s_0^i, a_0^i), \ldots, (s_{t-1}^i, a_{t-1}^i)\}$ of length t_i. We assume that human demonstrations reach the goal state and are optimal. The set of ground-truth constraints $s_t \setminus s_{t-1}$ and $s_{t-1} \setminus s_t$ become the supervision samples to train our $f_\theta(.)$ function as a neural model and generate learned parameters θ^*. Akin to a typical *divide-and-conquer* solution, this decomposition exemplifies our efforts to simplify the end-to-end problem of reaching goal states.

4 Technical Approach

We learn to predict the next robot action a_t, given world state s_t, instruction l and constraint history η_t. Assuming a given planner ψ, we realize the GoalNet neural model for goal-constraint prediction as follows (see Fig. 2):

$$\delta_t^+, \delta_t^- = f_\theta(s_t, l, \eta_t) = f_\theta^{goal}\left(f_\theta^{task}\left(f_\theta^{state}(s_t), l\right), f_\theta^{hist}(\eta_t)\right).$$

To do this, we encode the world state in the form of an object-centric graph. The state encoding is generated by fusing the relational and state information of the objects in the

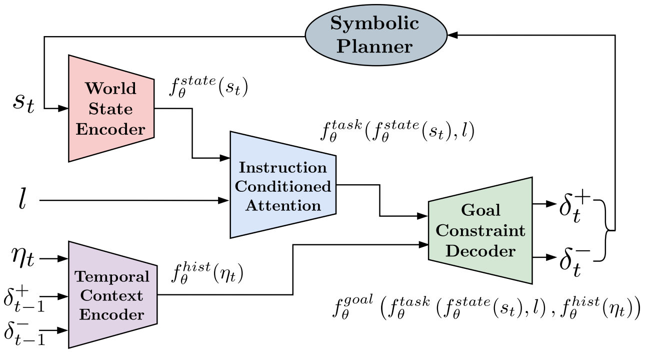

Symbolic Planner
s_t
World State Encoder
$f_\theta^{state}(s_t)$
$f_\theta^{task}(f_\theta^{state}(s_t), l)$
Instruction Conditioned Attention
l
Goal Constraint Decoder
δ_t^+
δ_t^-
η_t
δ_{t-1}^+
δ_{t-1}^-
Temporal Context Encoder
$f_\theta^{hist}(\eta_t)$
$f_\theta^{goal}\left(f_\theta^{task}\left(f_\theta^{state}(s_t), l\right), f_\theta^{hist}(\eta_t)\right)$

Figure 2: Using a symbolic planner and GOALNET in tandem.

environment using the function $f_\theta^{state}(.)$. We encode the constraint history using the function $f_\theta^{hist}(.)$. We then attend over the state encoding conditioned on the input task instruction where the attention weights are generated via $f_\theta^{task}(.)$. Finally, the positive and negative constraints to be established at time step t are predicted by the function $f_\theta^{goal}(.)$. The predicted constraints are then sent to the planner to generate an action a_t, executed in symbolic realizations of the environment and transition function and reach the next state s_{t+1}. We execute the above iteratively to reach a goal state.

At each time step t, we encode the current world state s_t as an object-centric graph $G_t = (\mathcal{O}(s_t), \mathcal{R}(s_t))$ where each node represents an object $o \in \mathcal{O}(s_t)$. Each relation $r \in \mathcal{R}(s_t)$ of the form $R(o^1, o^2)$ is encoded as a directed edge from o^1 to o^2 with edge type as R. To represent the states of an object o, we generate a binary vector $q_o = \{0, 1\}^u$ that represents the discrete object states for each of u states that include $\mathrm{Open/Closed}$, $\mathrm{On/Off}$, etc. Similarly, we generate a binary vector $p_o = \{0, 1\}^v$ that represents the presence of various object properties (1 if present and 0 otherwise) for each of v properties that include $\mathrm{isSurface}$ and $\mathrm{isContainer}$. We also incorporate a function $\mathcal{C}(.)$ that generate a dense vector representation for an input token of an object. For an object o, we represent this by $e_o = \mathcal{C}(o) = \mathbb{R}^w$ as a w-dimensional embedding. Our approach is agnostic to this function, but assumes that representations of semantically similar objects (such as with tokens "apple" and "orange") appear close, whereas semantically different objects appear far apart (such as with tokens "fork" and "table") in the learned space (Mikolov et al. 2018). Unless stated otherwise, we utilize $\mathrm{ConceptNet}$ embeddings (Speer, Chin, and Havasi 2019) to facilitate generalization (Tuli et al. 2021).

World State Encoder. We concatenate the embeddings q_o, p_o and e_o for each object $o \in \mathcal{O}(s_t)$ to form the object attributes that initialize each node of the graph representation of s_t. The relations of each object o in the edges $\mathcal{R}(s_t)$ is represented as an adjacency vector r_o. The relational information is first encoded using a d-layer Fully Connected Network (FCN) with Parameterized ReLU (PReLU) (He et al. 2015) activation as:

$$
\begin{aligned}
r_o^0 &= \mathrm{PReLU}(W_{rel}^0[r_o] + b_{rel}^0), \\
r_o^k &= \mathrm{PReLU}(W_{rel}^k[r_o^k] + b_{rel}^k), \\
r_o^d &= \mathrm{Sigmoid}(r_o^{d-1}),
\end{aligned}
\tag{3}
$$

where k varies from 0 to $d-1$, W_{rel}^k and b_{rel}^k represent the weight and bias parameters of the FCN. This results in a relational embedding for each object o as r_o^d. Now, we fuse the semantic and relational embeddings to generate an embedding of each object o as $[q_o, p_o, e_o, r_o^d]$. Late fusion of the relational information enables the downstream predictors to exploit the semantic and relational information independently, improving inference performance as demonstrated by prior work (Bansal et al. 2020). Thus, the output of our state-encoder becomes

$$
f_\theta^{state}(s_t) = \{\tilde{s}_t^o = [p_o, q_o, e_o, r_o^d] \mid \forall o \in \mathcal{O}(s_t)\}. \tag{4}
$$

Temporal Context Encoder. To ensure a seamless execution of the policy, it is crucial that the model is informed of the local context indicating the set of objects that the agent may manipulate in the future. This is typical in many real-life navigation and manipulation tasks where the sequential actions are temporally correlated. For instance, in the task of placing pillows on the couch, the agent first moves towards a pillow, grasps it and then places it on the couch. This entails that the sequence of goal constraints would initially have $\mathrm{Near}(\mathrm{robot}, \mathrm{pillow}_0)$, followed by $\mathrm{ConnectedTo}(\mathrm{pillow}_0, \mathrm{robot})$ and then $\mathrm{OnTop}(\mathrm{pillow}_0, \mathrm{couch}_0)$. This example demonstrates the high correlation between the interactions and manipulated objects of two adjacent time steps. Formally, GOALNET encodes the temporal history of the goal-constraints η_t using a Long-Short-Term-Memory (LSTM) neural model. For each relation $r = R_{t-1}(o_{t-1}^1, o_{t-1}^2) \in \mathcal{R}(s_{t-1})$ predicted in the previous time step $t-1$ in $\delta_{t-1}^+ \cup \delta_{t-1}^-$, we define an encoding $\tilde{r} = [\vec{R}_{t-1}, \mathcal{C}(o_{t-1}^1), \mathcal{C}(o_{t-1}^2)]$, where $\vec{R}_{t-1}$ is defined as a one-hot encoding for the relation type $R_{t-1} \in \mathcal{S}$ of the form $\{0, 1\}^{|\mathcal{S}|}$. $\mathcal{C}(o_{t-1}^1)$ and $\mathcal{C}(o_{t-1}^2)$ are the dense embeddings of the tokens of objects o_{t-1}^1 and o_{t-1}^2. At each time step t, we denote the encoding of the constraints history η_t as $\tilde{\eta}_t$ where

$$
f_\theta^{hist} = \tilde{\eta}_t = \mathrm{LSTM}([\vec{R}_{t-1}, \mathcal{C}(o_{t-1}^1), \mathcal{C}(o_{t-1}^2)], \tilde{\eta}_{t-1}).
$$

Instruction Conditioned Attention. The input language instruction l is encoded using a sentence embedding model represented as $\mathcal{B}(.)$ to generate the encoding $\tilde{l} = \mathcal{B}(l)$. We use the language instruction encoding $\tilde{l}$ to attend over the world objects using the following attention mechanism

$$
\begin{aligned}
\alpha_o &= \mathrm{Sigmoid}(W_{attn}[\tilde{s}_t^o, \tilde{l}] + b_{attn}), \ \forall o \in \mathcal{O}(s_t), \\
\tilde{s}_t^l &= \sum_{o \in \mathcal{O}(s_t)} \alpha_o \cdot \tilde{s}_t^o, \\
\tilde{s}_t &= \mathrm{PReLU}(W_{task}^k[\tilde{s}_t^l] + b_{task}^k)
\end{aligned}
\tag{5}
$$

Using a tokenizer (Bird, Klein, and Loper 2009), we also extract the set of objects in the instruction l and denote this set as O_l. We then generate the encoding of the objects in the instruction using Bahdanau style state-conditioned self-attention (Bahdanau, Cho, and Bengio 2014) as

$$
\begin{aligned}
\epsilon_o &= \mathrm{Sigmoid}(W_l[\mathcal{C}(o), \tilde{s}_t] + b_l), \ \forall o \in O_l, \\
\tilde{l}_{obj} &= \sum_{o \in O_l} \epsilon_o \cdot \mathcal{C}(o).
\end{aligned}
\tag{6}
$$

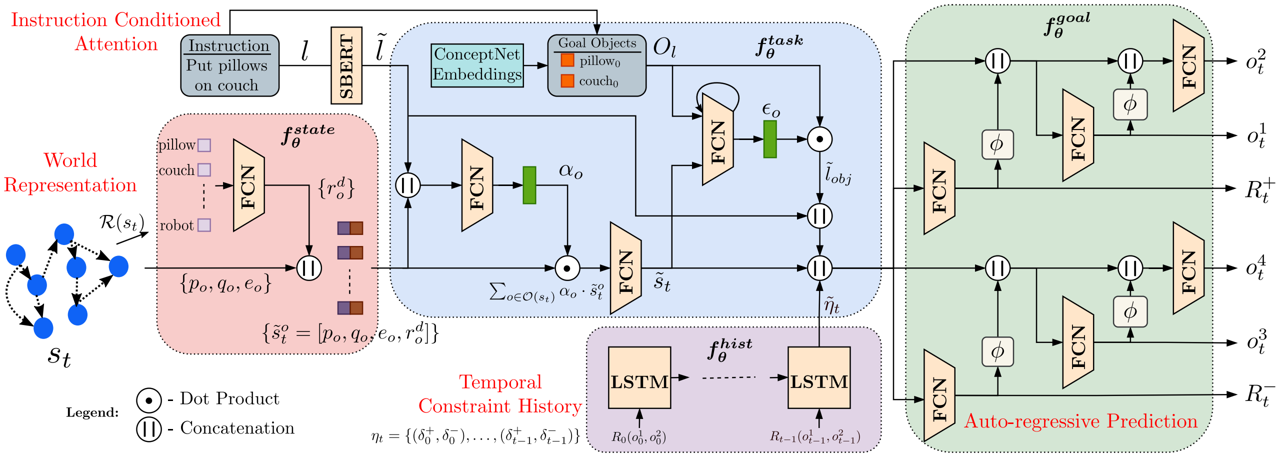


Figure 3: Details of the (colored) blocks from Figure 2. GOALNET neural model encodes the world state and uses goal information with the natural language instruction input to attend over a task-specific context, finally decoding the next symbolic constraints to send to an underpinning planner.

This gives the output of $f_\theta^{task}(.)$ as

$$f_\theta^{task}(\tilde{s}_t^o, l) = [\tilde{s}_t, \tilde{l}_{obj}, \tilde{l}]. \tag{7}$$

The attention operation aligns the information of the input language sentence with the scene to learn task-relevant context by allocating appropriate weights to objects. This relieves the downstream predictors from calculating the relative importance of world objects and focusing only on the ones related to the task, allowing the model to scale with the number of objects in the world.

Goal Constraint Decoder. GOALNET takes the instruction attended world state $\tilde{s}_t$, encoding of the constraint-history $\tilde{\eta}_t$, instruction objects encoding $\tilde{l}_{obj}$ and the sentence encoding $\tilde{l}$ to predict a pair of positive and negative constraints as relations $R_t^+(o_t^1, o_t^2)$ and $R_t^-(o_t^3, o_t^4)$. To predict each of the three components of relation and the two objects, we predict the likelihood score for each relation type $\mathcal{S}$ and objects $\mathcal{O}(s_t)$. We then select the relation or object with the highest likelihood scores. The three components are also predicted in an auto-regressive fashion. For instance, to predict $R_t^+(o_t^1, o_t^2)$, we first predict the relation R_t^+ and give the likelihood scores to the predictor for the first object o_t^1. Similarly, we forward likelihood scores of both R_t^+ and o_t^1 to predict object o_t^2. To do this, instead of using an argmax of the likelihood vector, we forward the Gumbel-Softmax of the vector (Jang, Gu, and Poole 2017) (denoted by $\phi(.)$). It is a variation of softmax function that allows us to generate a one-hot vector while also allowing gradients to backpropagate (as argmax is not differentiable). It uses a temperature parameter τ that allows to control how close the output is to one-hot versus the softmax output. We use a small constant value as τ in our model. Thus, we generate the likelihood scores using the following mechanism

$$\begin{aligned}
\tilde{R}_t^+ &= \text{softmax}(W_R^+[\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}] + b_R^+), \\
\tilde{o}_t^1 &= \text{softmax}(W_1^+[\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}, \phi(\tilde{R}_t^+)] + b_1^+), \\
\tilde{o}_t^2 &= \text{softmax}(W_2^+[\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}, \phi(\tilde{R}_t^+), \phi(\tilde{o}_t^1)] + b_2^+).
\end{aligned}$$

Then the predicate relation R_t^+ becomes $\text{argmax}_{R \in \mathcal{S}} \tilde{R}_t^+$, the first object o_t^1 is $\text{argmax}_{o \in \mathcal{O}(s_t)} \Omega(\tilde{o}_t^1, R_t^+)$ and o_t^2 as $\text{argmax}_{o \in \mathcal{O}(s_t)} \Omega(\tilde{o}_t^2, R_t^+)$. A similar mechanism is followed to predict the negative constraint $R_t^-(o_t^3, o_t^4)$. Here, Ω denotes masks that impose grammar constraints at inference time based on pre-conditions that the relations impose. The masks are used to force the likelihood scores of infeasible objects to 0. For instance, the OnTop relation only accepts the objects as the second argument that have the isSurface property. We also mask out the likelihood scores of $\tilde{o}_t^2$ and $\tilde{o}_t^4$ based on whether R_t^+ and R_t^- are constraints on the state of the objects in the first argument. To enable lexical action generation using a single positive and a single negative constraint, we use an anchored verb lexicon with post-conditions as described by Misra et al. (2015). Thus, the predicted relations are represented as

$$\begin{aligned}
R_t^+(o_t^1, o_t^2), R_t^-(o_t^3, o_t^4) &= f_\theta^{goal}(\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}), \\
\delta_t^+, \delta_t^- &= \{R_t^+(o_t^1, o_t^2)\}, \{R_t^-(o_t^3, o_t^4)\}.
\end{aligned} \tag{8}$$

The model is trained using a loss *viz* the sum of binary cross-entropy with ground-truth predicates for the six predictors.

Symbolic Planner. These constraint sets generated by the GOALNET model are passed on to a planner ψ that generates an action sequence as $\psi(s_t, \delta_t^+, \delta_t^-, \Lambda)$, that is executed by the agent. All pre-conditions and effects are encoded using a symbolic Planning Domain Definition Language (PDDL) file and is denoted as Λ. Unlike prior work that uses imitation or reinforcement learning (Tuli et al. 2021; Bae et al. 2019), the PDDL information allows the planner to utilize symbolic information, indirectly informing the neural model. After running the planner, the executed action sequence and resulting world state is provided as input to the model for predicting the goal constraints and subsequently the action in the next step. Given such a planner, for every state s_j^i, we generate the goal-standard predicates using a single-step difference over the sets of relations between the two consecutive states as

$$\begin{aligned}
\hat{\delta}_j^+, \hat{\delta}_j^- &= s_{j+1}^i \setminus s_j^i, s_j^i \setminus s_{j+1}^i \forall j < t_i - 1, \\
\hat{\delta}_{t_i-1}^+, \hat{\delta}_{t_i-1}^- &= \emptyset, \emptyset.
\end{aligned}$$

We encode the ground-truth set of predicates as binary vectors of relations and two objects. We then use the loss function to train the neural model in a teacher forced manner (Toomarian and Barhen 1992). However, at inference time, as we do not have ground-truth labels, we need to feed back the next state through emulation/planning, causing the model to be biased to the exposure and conditioning of the training data. To alleviate the effects of this bias, we use planners at two levels of fidelity: a *low-fidelity* simulator and a *high-fidelity* symbolic planner. In training, with a constant probability of p, we utilize a low-fidelity *symbolic simulator* (referred to as SymSim in the rest of the discussion) to train the model for each datapoint. SymSim emulates the effects of generating and executing action corresponding to the predicted goal predicates δ_t^+, δ_t^- to generate the next state

$$\hat{s}^i_{j+1} = s^i_j \cup \delta^+_t \setminus \delta^-_t, \tag{9}$$

where the $\cup$ and $\setminus$ operations are performed on the relation set of the graph s^i_j to generate a new graph $\hat{s}^i_{j+1}$. In lieu of using a high-fidelity symbolic planner to generate actions and update the world state, this enables us to reduce training time significantly. In training, the recurrent predicate prediction stops when $\delta^+_t \cup \delta^-_t = \emptyset \vee t \geq t_i$, which we use as a proxy in lieu of explicitly learning the goal-check function.

On the other hand, at test time, we use a high-fidelity symbolic planner to generate the subsequent state as well as an action sequence at every time step t. The execution stops when $\delta^+_t \cup \delta^-_t = \emptyset \vee t \geq 30$, where the upper bound for recurrent execution is kept to thirty actions, *viz*, the maximum plan length in the training dataset. The symbolic planner we use for evaluation is referred to as Rintanen and is taken from Rintanen (2012). This symbolic planner is able to predict actions and update the environment state with action effects and side effects, providing a complete update of the world state. We denote the updated state for an input datapoint with initial state s^i_0 as $\bar{s}^i_1$ and subsequent states as $\bar{s}^i_j$ where j varies from 2 to T_i with the length of the action sequence generated by GoalNet is T_i. We similarly represent generated actions by $\bar{a}^i_j \; \forall j \in \{0, \ldots, T_i - 1\}$.

Word and Sentence Embeddings. GoalNet uses word embedding function $\mathcal{C}(\cdot)$ that provides a dense vector representation for word tokens associated with object class types. Word embeddings represent words meanings in a continuous space where vectors close to each other are semantically related, facilitating model generalization to novel object types encountered online. Using such (pre-trained) embeddings incorporates *general purpose* object knowledge to facilitate richer generalization for downstream predicate learning. Similarly, we use SentenceBERT as our $\mathcal{B}(.)$ function that encodes the natural language instruction sentence (Reimers et al. 2019). This is a modification of the pre-trained BERT model (Devlin et al. 2019) that uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings. The complete model is summarized in Fig. 3.

5 Evaluation and Results

Data set Description. To demonstrate the efficacy of the GoalNet model, we utilize the dataset made available by Misra et al. (2015). The dataset consists of natural language instructions as a sequence of sentences, state, and action sequences collected through crowd-sourcing. The dataset is collected from two domains: kitchen and living room, each containing 40 objects, each with up to 4 instances of each object class. The dataset consists of diverse instruction types ranging from short-horizon tasks such as *"go to the sink"* to high-level tasks involving complex and multiple interactions with environment objects such as *"cook ramen in a pot of water"*. See Table 1 for a list of sample objects in the domain, their possible states and properties with the set of symbolic actions allowed to be executed.

Objects:	(*kitchen*) sink, stove, mug, microwave, fridge, icecream, kettle, coke, plate, boiledegg, salt, stovefire, sinkknob. (*living room*) loveseat, armchair, coffeetable, tv, pillow, bagofchips, bowl, garbagebag, shelf, book, coke, beer.
Object States:	Open/Closed, On/Off, HasWater/HasChocolate/IsEmpty, DoorOpen/DoorClosed.
Object Properties:	IsSurface, IsTurnable, IsGraspable, IsPressable, IsOpenable, IsSqueezeable, IsContainer.
Actions:	Grasp, Release, MoveTo, PlaceOn, PlaceIn, Press, Pour, Squeeze, stateOn, stateOff, stateOpen, stateClose.
Predicates:	OnTop, Near, ConnectedTo.

Table 1: Sample set of objects, states, properties, actions and predicates (for complete lists, see Table 4)

We extract each instruction sentence from a sequence of language instructions (She and Chai 2016). We also extract the action sequence and environment states after each language instruction is executed in the environment. Thus, for each natural language instruction l^i, we have a state-action sequence $\{(s^i_0, a^i_0), \ldots, (s^i_{t_i-1}, a^i_{t_i-1})\}$ with the plan length of t_i. The dataset consists of 1117 such data points, where we use a split of $70\% : 15\% : 15\%$ datapoints for our train, validation and test datasets as per prior work (She and Chai 2016). We also augment the training data to generate additional by perturbing the training data points by semantic object replacement in both states $\{s^i_j\}_j$ and actions $\{a^i_j\}_j$ as per the ConceptNet embeddings (Tuli et al. 2021). We perform this only for objects and datapoints in the training and validation sets wherein the replaced object is *unseen* in the original data. This allowed us to increase the number of training datapoints by 25%, giving a total of 633 unique starting states (object-centric graphs) in the dataset, facilitating more diverse supervision to GoalNet. The test set consists of initial states distinct from those in the training to ensure a fair and robust evaluation of predicate prediction.

Baseline and Evaluation Metrics. We adopt the approach by She and Chai (2016) as the baseline. Building on the primitives developed by Misra et al. (2015), this work maps closely to our setting where the objective maps directly to the prediction of goal predicates to be passed onto a symbolic planner. This approach addresses the problem of inferring a hypothesis space for verb frames extracted from language instructions. The determination of hypothesis space uses heuristics to focus the learner only on a focused set of predicates. Further, hand-crafted features extracted from language instruction, world state and the candidate goal states is used during learning. A log-linear model is then trained incrementally with the successful demonstration of a plan. We adopt this model as a baseline and compare the

Model	**SJI**	**IED**	**F1**	**GRR**
Baseline (She and Chai 2016)	0.448	0.450	0.512	0.370
GoalNet	0.562	0.601	0.651	0.468
Model Ablations				
- Relational information	0.533	0.575	0.621	0.449
- Instance grounding	0.528	0.567	0.617	0.439
- δ^- prediction	0.424	0.447	0.533	0.416
- δ^+ prediction	0.156	0.167	0.195	0.098
- Temporal context encoding	0.221	0.323	0.263	0.159
- Goal Object Attn	0.547	0.595	0.634	0.430
- Instruction conditioned Attn	0.565	0.604	0.650	0.456
- Grammar mask	0.567	0.602	0.651	0.459
Model Explorations				
Instruction encoding : Conceptnet	0.389	0.451	0.477	0.297
Temporal Context $(\delta^+_{t-1} \cup \delta^-_{t-1})$	0.552	0.580	0.637	0.430
Temporal Context (s_{t+1})	0.503	0.553	0.593	0.380
Training using Rintanen	**0.645**	**0.661**	**0.717**	**0.558**

Table 2: A comparison of goal-prediction and goal-reaching performance for the baseline, the proposed GoalNet model, ablations and explorations. Results are presented for test set derived from living room and kitchen domains.

approach in a batch (instead of an incremental) setting.

The accuracy of the goal-predicate predictions is evaluated as the ability to reach goal states using such predicates. We use popular metrics to evaluate the model (She and Chai 2016). We define the aggregate goal-predicates for a data point $\{(s^i_0, a^i_0), \ldots, (s^i_{t_i-1}, a^i_{t_i-1})\}$ and model generated sequence $\{(s^i_0, \bar{a}^i_0), \ldots, (\bar{s}^i_{T_i-1}, \bar{a}^i_{T_i-1})\}$ as

$$\hat{\Delta}^+_i, \hat{\Delta}^-_i = s^i_{t_i-1} \setminus s^i_0, s^i_0 \setminus s^i_{t_i-1},$$
$$\Delta^+_i, \Delta^-_i = \bar{s}^i_{T_i-1} \setminus s^i_0, s^i_0 \setminus \bar{s}^i_{T_i-1}.$$

Thus, for a dataset of size N, we define

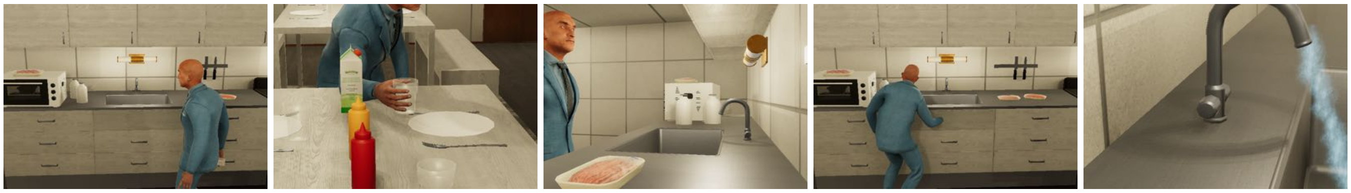
- *SJI (State Jaccard Index)* checks the overlap between the aggregate predicates as
$$\text{SJI} = \frac{1}{N} \sum_{i=1}^{N} \frac{\|\hat{\Delta}^+_i \cap \Delta^+_i\| + \|\hat{\Delta}^-_i \cap \Delta^-_i\|}{\|\hat{\Delta}^+_i \cup \Delta^+_i\| + \|\hat{\Delta}^-_i \cup \Delta^-_i\|}.$$
- *IED (Instruction Edit Distance):* measures the overlap between the similarity between the generated action sequence $\{\bar{a}^i_0, \ldots, \bar{a}^i_{T_i-1}\}$ and ground-truth sequence $\{a^i_0, a^i_{t_i-1}\}$. Specifically, the edit distance d^i between these two sequences is used as
$$\text{IED} = \frac{1}{N} \sum_{i=1}^{N} 1 - \frac{d^i}{\max(T_i, t_i)}.$$
- *GRR (Goal Reaching Rate)* evaluates if the aggregated ground-truth predicates are present in the predicted ones
$$\text{GRR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\Delta^+_i \subseteq \hat{\Delta}^+_i \wedge \Delta^-_i \subseteq \hat{\Delta}^-_i).$$
- *F1* evaluates the average of the F1 scores between the positive and negative aggregate predicate sets.

Baseline Comparison Table 2 presents the scores for the baseline, GoalNet and its variations. For a fair comparison with baseline, we have considered a grounding-aware version of GoalNet where we include the instance groundings for all objects in the input instruction as part of the goal-object set O_l (see Section 4). GoalNet improves SJI, IED, F1 and GRR by 25%, 34%, 27% and 26%, respectively. However, when we use Rintanen instead of SymSim, we get marked improvement, albeit taking $13\times$ more time to train the model. With Rintanen in the training loop, we get 44%, 47%, 40% and 51% higher SJI, IED, F1 and GRR scores demonstrating the importance of action side-effects introduced by the planner in lieu of using a simplified simulator. The improvement is primarily due to the dense representation of the input state, unlike the hand-crafted feature approach of She and Chai (2016), enabling GoalNet to generalize to settings unseen at training. Figure 4 and 5 shows state-action pair generated by GoalNet in kitchen and living room domain respectively, demonstrating its ability to execute tasks successfully and reach a goal state.

Analysis of Model Components Table 2 also presents scores corresponding to model ablations. We fix the model capacities for a fair comparison. Without the relational information in the form of adjacency matrix $\mathcal{R}(s_t)$ for input s_t, the model is unable to capture change in the spatial relations among world objects. For instance, when filling a mug with water (see Fig. 4), the $\text{PlaceOn}(\text{mug}_0, \text{sink}_0)$ establishes a OnTop relation between the two objects. Missing such changes in the state lead to a drop in performance of $\sim 5\%$. Without the instance grounding of the objects the problem becomes harder. For instance, without knowing that *coffee-table* in the instruction *"place beer on top of coffee-table"* is mapped to table_0, the agents needs to additionally infer the specific instance that needs to be manipulated in case multiple tables are present (see Fig. 5). We see a drop of ∼6% in this case, though still higher than the baseline.

An alternative approach could be to predict only positive or negative predicates. For instance, when we only predict positive predicates (- δ^- *prediction*), GRR drops by ∼11%. Such a model is unable to predict the required predicates when only relations are removed from the state. Examples include dropping objects when negative predicates include $\text{Connected}(\text{robot}, \text{o})$ for an object o. Similarly, when only predicting the negative predicates (- δ^+ *prediction*), the model suffers a drop of ∼80% in goal reaching performance. This evidence demonstrates the importance of predicting both positive and negative predicates to be sent to the underpinning planner/simulator. The inclusion of temporal context allows learning of correlated and commonly repeated action sequences. For instance, the task of filling a mug with water typically involves placing the mug beneath a faucet/tap and turning on the tap (see Fig. 4). The ablation of this component leads to erroneous predictions when a particular predicate in a common plan fragment is missing or incorrectly predicted, for instance, when turning the tap on without placing the mug underneath.

Successful execution of an instruction may involve manipulation of multiple objects, such as beer and wine, when fetching both (see Fig. 5). Attention over the objects in the input language instruction (see eq. 6) enables GoalNet to attend over the goal objects to dynamically prioritize manipulation at each time step. When we replace this (- *Goal Object Attn*) with the mean of ConceptNet embeddings of goal objects, the GRR drops by ∼8%. We use language in-



Initial State
Instr.: *"fill water in mug"*

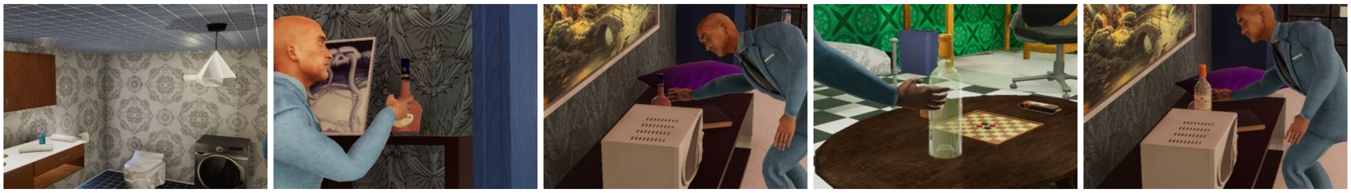
ConnectedTo(mug$_0$, robot)
{MoveTo(robot, mug$_0$), Grasp(robot, mug$_0$)}

Near(robot, sink$_0$)
{MoveTo(robot, sink$_0$)}

OnTop(mug$_0$, sink$_0$)
{PlaceOn(mug$_0$, sink$_0$)}

stateIsOn(tap$_0$)
{stateOn(tap$_0$)}

Figure 4: Sample plan in kitchen. Visualizations developed utilizing the VirtualHome Simulator (Puig et al. 2018) and a human-like agent with functionality akin to a single-arm manipulator. Predicted goal predicates shown in red. Executed plan at each time step shown in blue.



Initial State
Intr.: *"place beer and wine on top of the coffee-table"*

ConnectedTo(beer$_0$, robot)
{MoveTo(robot, beer$_0$), Grasp(robot, beer$_0$)}

OnTop(beer$_0$, table$_0$)
{MoveTo(robot, table$_0$), PlaceOn(beer$_0$, table$_0$)}

ConnectedTo(wine$_0$, robot)
{MoveTo(robot, wine$_0$), Grasp(robot, wine$_0$)}

OnTop(wine$_0$, table$_0$)
{MoveTo(robot, table$_0$), PlaceOn(wine$_0$, table$_0$)}

Figure 5: Sample plan in the living-room domain.

Model	**Verb Replacement**				**Paraphrasing**			
	SJI	**IED**	**F1**	**GRR**	**SJI**	**IED**	**F1**	**GRR**
Baseline	0.134	0.137	0.146	0.138	0.124	0.127	0.136	0.128
GoalNet	0.325	0.376	0.403	0.228	0.318	0.360	0.398	0.212

Table 3: GOALNET demonstrates the ability to generalize in case of *verb replacement* and *paraphrasing* relative to the baseline.

struction encoding $\tilde{l}$ to attend over the world objects. The attention operation aligns the information of the input language sentence with the scene to learn task-relevant context by allocating appropriate weights to objects. Without this, the GRR score drops by 2.6%. Finally, without the grammar mask Ω, the GRR drops marginally (2%), showing the ability to learn grammar-related semantics of the domain.

Model Explorations. We also explore additional variations of GOALNET. For instance, when encoding the language instruction using ConceptNet instead of SBERT, the scores drop by at least $\sim$26%. This highlights the power of pretrained language models in their ability to encode the task intention in natural language instructions. Additionally, we explore sending past predictions of both δ_t^+ and δ_t^- predicates when encoding the temporal context. In such a case, the GRR score drops by 8%. Similarly, when encoding temporal context using the encoding of the previous states, *i.e.*, utilizing $\tilde{s}_t$, we see a drop in GRR by 19%. This indicates that we need only the positive predicate information to encode the temporal context and additional information is superfluous to predict goal predicates effectively.

Generalization. We additionally test the ability of GOALNET to generalize to unseen instruction inputs by building two generalization data sets (see Table 3). We test the performance when replacing verb frames in the training data with those absent in the data set. For instance, we replace *boil* in *"boil milk"* with *heat*. Even in such cases, GOALNET is able to successfully reach the goal state (see Fig. 8). We additionally paraphrase the language input to test instruction level generalization. For example, we paraphrase *"gather all used plates and glasses, place into sink"* to *"collect all used dishes and glasses, keep in wash basin"*.

Table 3 above presents the performance scores of the baseline and GOALNET models. It is observed that the baseline is unable to generalize to unseen verb frames and objects without any human intervention. On the other hand, GOALNET generalizes in case of novel object references and unseen verbs relative to the baseline. GOALNET improves GRR by 65% in both cases of verb replacement and paraphrasing instructions. This generalization is achieved mainly due to the presence of dense token (ConceptNet) and instruction (SBERT) representations as opposed to storing observed verb-frame hypotheses in the baseline.

6 Conclusions

This paper proposes GOALNET, a novel neural architecture that learns to infer goal predicates for an input language instruction and world state, which when passed to an underpinning symbolic planner enables reaching goal states. GOALNET leverages independent inference over the objects in the world state and the spatial relations, applying instruction conditioned attention and using temporal contexts to autoregressively predict goal predicates. GOALNET is trained using human demonstrations in kitchen and living-room environments and is able to generalize to unseen settings. This work demonstrates how learning and classical planning can be tied together to address the challenge of following multistage tasks for a robot. The neural model enables generalization to unseen language instructions, outperforming a state-of-the-art baseline in terms of the goal reaching performance. Future work will investigate out of domain generalization to apriori unknown number of objects, learning from sub-optimal or failed plans, and principled handling ambiguity among equally plausible goals.

References

Bae, H.; Kim, G.; Kim, J.; Qian, D.; and Lee, S. 2019. Multi-Robot Path Planning Method Using Reinforcement Learning. *Applied Sciences*, 9(15): 3057.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baker, C. L.; Saxe, R.; and Tenenbaum, J. B. 2009. Action understanding as inverse planning. *Cognition*, 113(3): 329–349.

Bansal, R.; Tuli, S.; Paul, R.; and Mausam. 2020. TOOLNET: Using Commonsense Generalization for Predicting Tool Use for Robot Plan Synthesis. In *Workshop on Advances & Challenges in Imitation Learning for Robotics at Robotics Science and Systems (RSS)*.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Boularias, A.; Duvallet, F.; Oh, J.; and Stentz, A. 2015. Grounding spatial relations for outdoor robot navigation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 1976–1982. IEEE.

Bragg, J.; and Brunskill, E. 2020. Fake it till you make it: Learning-compatible performance support. In *Uncertainty in artificial intelligence*, 915–924. PMLR.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, 4171–4186.

Dragan, A. D.; Bauman, S.; Forlizzi, J.; and Srinivasa, S. S. 2015. Effects of robot motion on human-robot collaboration. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 51–58. IEEE.

Fitzgerald, T.; Goel, A.; and Thomaz, A. 2021. Modeling and Learning Constraints for Creative Tool Use. *Frontiers in Robotics and AI*, 8.

Gajewski, P.; Ferreira, P.; Bartels, G.; Wang, C.; Guerin, F.; Indurkhya, B.; Beetz, M.; and Śniezyński, B. 2019. Adapting everyday manipulation skills to varied scenarios. In *2019 International Conference on Robotics and Automation (ICRA)*, 1345–1351. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

Howard, T. M.; Tellex, S.; and Roy, N. 2014. A natural language planner interface for mobile manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 6652–6659. IEEE.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical reparameterization with gumbel-softmax. *International Conference on Learning on Learning Representations (ICLR)*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, A. X.; Gupta, A.; Lu, H.; Levine, S.; and Abbeel, P. 2015. Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5265–5272. IEEE.

Lesh, N.; and Etzioni, O. 1995. A sound and fast goal recognizer. In *IJCAI*, volume 95, 1704–1710. Citeseer.

Liao, Y.-H.; Puig, X.; Boben, M.; Torralba, A.; and Fidler, S. 2019. Synthesizing environment-aware activities via activity sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6291–6299.

Mann, J. 2021. *Neural Bayesian goal inference for symbolic planning domains*. Ph.D. thesis, Massachusetts Institute of Technology.

Mei, H.; Bansal, M.; and Walter, M. R. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 720–730.

Meneguzzi, F. R.; and Pereira, R. F. 2021. A Survey on Goal Recognition as Planning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI), 2021, Canadá*.

Mikolov, T.; Grave, É.; Bojanowski, P.; Puhrsch, C.; and Joulin, A. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Misra, D.; Bennett, A.; Blukis, V.; Niklasson, E.; Shatkhin, M.; and Artzi, Y. 2018. Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2667–2678.

Misra, D.; Tao, K.; Liang, P.; and Saxena, A. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 992–1002.

Misra, D. K.; Sung, J.; Lee, K.; and Saxena, A. 2016. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35(1-3): 281–300.

Pflueger, M.; and Sukhatme, G. S. 2015. Multi-step planning for robotic manipulation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2496–2501. IEEE.

Puig, X.; Ra, K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8494–8502.

Reimers, N.; Gurevych, I.; Reimers, N.; Gurevych, I.; Thakur, N.; Reimers, N.; Daxenberger, J.; Gurevych, I.;

Reimers, N.; Gurevych, I.; et al. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 671–688. Association for Computational Linguistics.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193: 45–86.

She, L.; and Chai, J. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 108–117.

Shridhar, M.; Thomason, J.; Gordon, D.; Bisk, Y.; Han, W.; Mottaghi, R.; Zettlemoyer, L.; and Fox, D. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10740–10749.

Speer, R.; Chin, J.; and Havasi, C. 2019. ConceptNet Numberbatch, the best pre-computed word embeddings you can use. *GitHub repository*.

Squire, S.; Tellex, S.; Arumugam, D.; and Yang, L. 2015. Grounding English commands to reward functions. In *Robotics: Science and Systems*.

Suhr, A.; and Artzi, Y. 2018. Situated Mapping of Sequential Instructions to Actions with Single-step Reward Observation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2072–2082.

Thomason, J.; Zhang, S.; Mooney, R. J.; and Stone, P. 2015. Learning to interpret natural language commands through human-robot dialog. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Toomarian, N. B.; and Barhen, J. 1992. Learning a trajectory using adjoint functions and teacher forcing. *Neural networks*, 5(3): 473–484.

Tuli, S.; Bansal, R.; Paul, R.; and , M. 2021. TANGO: Commonsense Generalization in Predicting Tool Interactions for Mobile Manipulators. In *International Joint Conference on Artificial Intelligence, IJCAI-21*, 4197–4205.

A Dataset and Domain details

Domain Details. A typical world state of kitchen and living room environment consists of 40 objects each (Table 4). Some objects in the dataset do not play any functional role and have been removed to facilitate training. These include buttons of the Fridge and Microwave in the kitchen domain and buttons of the TV remote in the scenes from the living room domain. Each object has an instance identifier $(\mathrm{mug}_1, \mathrm{mug}_2)$, a set of properties such as $\mathrm{IsGraspable}$ and $\mathrm{IsPourable}$ used for planning and a set of boolean states such as $\mathrm{HasWater}/\mathrm{HasCoffee}$ that can be changed by robot actions. The robot is also an object in the environment. The environment model consist of 12 actions including Grasp, MoveTo and Pour, each having environment objects as arguments. Examples include $\mathrm{Grasp}(\mathrm{robot}, \mathrm{mug}_1)$ and $\mathrm{Pour}(\mathrm{coke}_0, \mathrm{mug}_1)$. Each action results in effects encoded as postconditions in the domain description (via PDDL). The encoded effects take the form of conjunction of predicates or their negations. Action can introduce a new spatial relation between two objects, for example, $(\mathrm{OnTop}(\mathrm{table}_0, \mathrm{book}_0)$ or modify state of object, such as $\mathrm{State}(\mathrm{kettle}, \mathrm{HasWater})$.

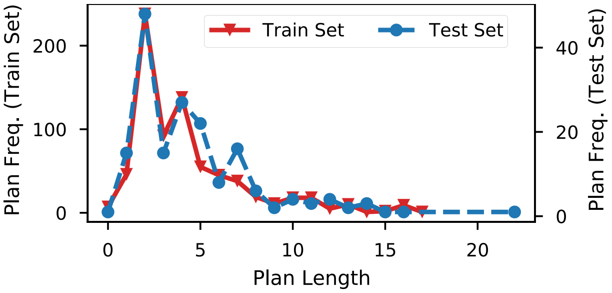


Figure 6: Frequency of plans with plan length.

Dataset Details. The dataset is devoid of the metric information such as position as well as the point-cloud models that includes the object geometries, bounding boxes and pose data. Both training and test datasets consist of long-horizon plans with up to 30 action sequences, albeit with decreasing frequency as we increase plan length (see Fig. 6). The original dataset of Misra et al. (2015) defines ten high-level objectives, five for each domain. These include *"make ramen"*, *"clean the room"*, *"make coffee"*, *"prepare room for movie night"*, etc. Each high level task is described as a sequence of low level instructions mapped to an initial environment and action sequence. For instance, high level task of *"make ramen"* is decomposed as *"Take pot on counter and fill it with water from the sink"*. *"Place the pot on a burner on the stove"*. *"Turn on the burner and let the water boil"*. *"Once it is boiling, open the lid of the ramen cup and pour boiling water in until it reaches the top of the container"*. She and Chai (2016) extract these low level instructions, each with a single verb and its arguments. The above decomposition leads to a training set with 77 unique verbs, an average of 6 words per instruction text and average plan length for train and test instances of ∼5. Out of 1117 data instances, 56% are from kitchen domain and remaining from living room. We perform data cleaning as performed by She and Chai (2016) to remove noise in the dataset, for instance, removing *wait* statements in the discrete-time control setup we consider.

B Implementation and Training Details

We detail the hyper-parameters for the GoalNet architecture introduced in this paper. The Parameterized ReLU activation function with a 0.25 negative input slope was used in all hidden layers of the world-state encoder described in eq. 3. The word embeddings (derived from $\mathrm{ConceptNet}$) have a size of 300. Sentence embeddings from SBERT have a size of 384. Additionally, the properties of objects such as $\mathrm{isGraspable}$, $\mathrm{isContainer}$, etc. is encoded as a one-hot vector of size 12. Object states such as $\mathrm{Open}/\mathrm{Close}$ are also encoded as one-hot vectors of size 7.

- *World-State Encoder:* The relational information between object nodes, in the form of adjacency matrix, is

Objects in Kitchen : robot, counter$_1$, sink, stove, mug$_1$, microwavedoor, microwave, fridge, fridgeleftdoor, fridgerightdoor, spoon$_1$, icecream$_1$, kettle, ramen$_1$, syrup$_1$, glass$_1$, longcup$_1$, longcup$_2$, fork$_1$, energydrink$_1$, coke$_1$, canadadry$_1$, plate$_1$, plate$_2$, syrup$_2$, instantramen$_1$, boiledegg$_1$, salt$_1$, light$_1$, stoveknob$_1$, stoveknob$_2$, stoveknob$_3$, stoveknob$_4$, stovefire$_1$, stovefire$_2$, stovefire$_3$, stovefire$_4$, fridgebutton, microwavebutton, sinkknob, icecreamscoop.

Objects in Living Room: icecreamscoop, robot, loveseat$_1$, armchair$_1$, armchair$_2$, coffeetable$_1$, tvtable$_1$, tv$_1$, tv$_1$remote$_1$, pillow$_1$, pillow$_2$, pillow$_3$, snacktable$_1$, bagofchips$_1$, bowl$_1$, garbagebag$_1$, garbagebin$_1$, shelf$_1$, shelf$_2$, book$_1$, book$_2$, coke$_1$, beer$_1$, xboxcontroller$_1$, xbox$_1$, cd$_2$, cd$_1$, tv$_1$powerbutton, tv$_1$channelupbutton, tv$_1$channeldownbutton, tv$_1$volumeupbutton, tv$_1$volumedownbutton, tv$_1$remote$_1$powerbutton, tv$_1$remote$_1$channelupbutton, tv$_1$remote$_1$channeldownbutton, tv$_1$remote$_1$volumeupbutton, tv$_1$remote$_1$volumedownbutton, tv$_1$remote$_1$mutebutton.

Object States: Open/Closed, On/Off, DoorOpen/DoorClosed, HasIceCream/HasEgg/HasRamen/HasCoffee/HasCD/HasWater, HasChips, HasSpoon/HasVanilla/HasChocolate/IsEmpty/IsScoopsLeft, OnChannel1/OnChannel2/OnChannel3/OnChannel4, VolumeUp/VolumneDown.

Object Properties: IsAddable, IsScoopable, isSurface, IsGraspable, isPressable, IsTurnable, IsSqueezeable, isContainer,isOpenable

Actions: Grasp, Release, MoveTo, PlaceOn, PlaceIn, Press, Pour, Squeeze, stateOn, stateOff, stateOpen, stateClose.

Predicates: OnTop, Inside, Near, ConnectedTo.

Action Preconditions

$\mathrm{Grasp(robot, o) : IsGraspable(o) \wedge Near(robot, o) \wedge \neg ConnectedTo(o, robot)}$
$\mathrm{Release(robot, o) : ConnectedTo(o, robot)}$
$\mathrm{MoveTo(robot, o) : \emptyset}$
$\mathrm{PlaceOn(o^1, o^2) : ConnectedTo(o^1, robot) \wedge isSurface(o^2)}$
$\mathrm{PlaceIn(o^1, o^2) : ConnectedTo(o^1, robot) \wedge Near(o^2, robot) \wedge isContainer(o^1)}$
$\mathrm{Press(o^1) : Near(robot, o^1) \wedge isPressable(o^1)}$
$\mathrm{Pour(o^1, o^2) : ConnectedTo(o^1, robot) \wedge Near(o^2, robot)}$
$\mathrm{Squeeze(o^1, o^2) : ConnectedTo(o^1) \wedge Near(o^2, robot) \wedge IsSqueezable(o^1)}$
$\mathrm{StateOn(o) : Off(o) \wedge IsTurnable(o)}$
$\mathrm{StateOff(o) : On(o) \wedge IsTurnable(o)}$
$\mathrm{StateOpen(o) : Closed(o) \wedge IsOpenable(o)}$
$\mathrm{StateClose(o) : Open(o) \wedge IsOpenable(o)}$

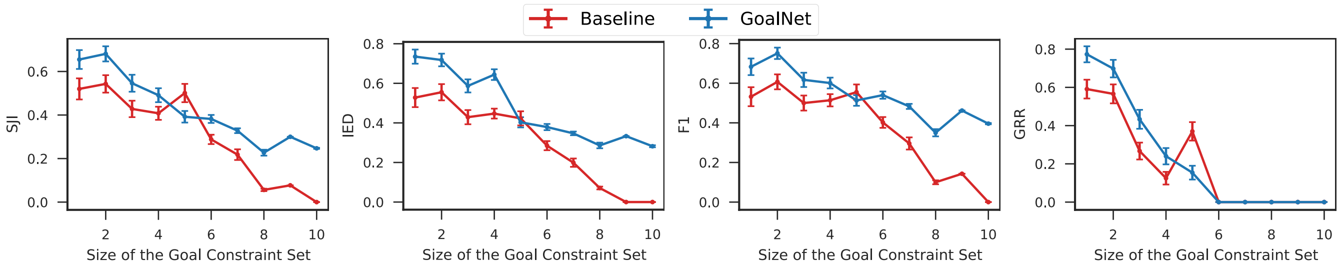
Actions Post-conditions

$\mathrm{Grasp(robot, o) : ConnectedTo(o, robot)}$
$\mathrm{Release(robot, o) : \neg ConnectedTo(o, robot)}$
$\mathrm{MoveTo(robot, o) : Near(o, robot)}$
$\mathrm{PlaceOn(o^1, o^2) : OnTop(o^1, o^2)}$
$\mathrm{PlaceIn(o^1, o^2) : Inside(o^1, o^2)}$
$\mathrm{Press(o^1) : On(o^1)}$
$\mathrm{Pour(o^1, o^2) : Inside(o^1, o^2)}$
$\mathrm{Squeeze(o^1, o^2) : \neg HasWater(o^1) \wedge \neg Inside(o^1, o^2)}$
$\mathrm{StateOn(o) : On(o)}$
$\mathrm{StateOff(o) : Off(o)}$
$\mathrm{StateOpen(o) : Open(o)}$
$\mathrm{StateClose(o) : Closed(o)}$

Table 4: Complete list of objects in kitchen and living room domains. Properties and states of objects. Predicates between objects. Actions with their pre-conditions and post-conditions.

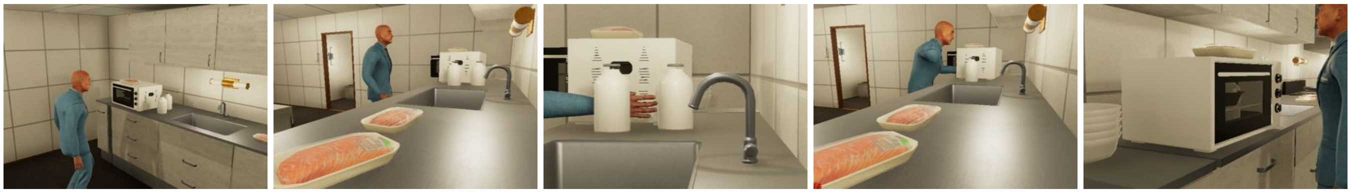
encoded using a 1-layer FCN of layer size $\mathcal{O}(s_t) \times 1$ with the sigmoid activation function.

- *Temporal Context Encoder:* A Long Short Term Memory (LSTM) layer of size 128 is used to encode the temporal history of predicted goal constraints, encoded as a concatenation of likelihood vectors as $[\tilde{R}_t^+, \tilde{o}_t^1, \tilde{o}_t^2]$.
- *Instruction Conditioned Attention:* The language instruction is embedded using a pretrained SBERT and then passed through a 1-layer FCN of output size 128. We attend over the state encoding conditioned on the encoded input task instruction where the attention weights for each object in the state are generated using a 1-layer FCN with the sigmoid activation function. Next, we use these goal-conditioned state object embeddings to generate the encoding of the objects in the instruction using Bahdanau style self-attention. This is again achieved using a 1-layer FCN with the sigmoid activation function, generating attention weights for each of the objects specified in the input instruction.
- *Goal Constraint Decoder.* After generating the final goal embedding by concatenating the instruction attended world state $\tilde{s}_t$, encoding of the constraint-history $\tilde{\eta}_t$, encoding of instruction objects $\tilde{l}_{obj}$ and the sentence encoding $\tilde{l}$, we predict the predicates. We pass $[\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}]$ through a 1-layer FCN of size 128 and PReLU activation function. We predict a pair of positive and nega-



(a) SJI Score (b) IED Score (c) F1 Score (d) GRR Score

Figure 7: Performance of baseline and GOALNET model with the size of aggregate goal-constraint sets.



Initial State
Instr.: *"heat milk"*

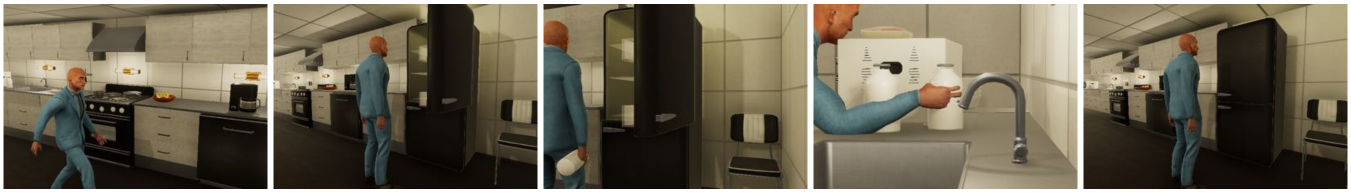
$\text{stateIsOpen}(\text{microwave}_0)$
$\{\text{MoveTo}(\text{robot}, \text{microwave}_0), \text{stateOpen}(\text{microwave}_0)\}$

$\text{ConnectedTo}(\text{milk}_0, \text{robot})$
$\{\text{MoveTo}(\text{robot}, \text{milk}_0), \text{Grasp}(\text{robot}, \text{microwave}_0)\}$

$\text{Inside}(\text{milk}_0, \text{microwave}_0)$
$\{\text{MoveTo}(\text{robot}, \text{microwave}_0), \text{PlaceIn}(\text{milk}_0, \text{microwave}_0)\}$

$\text{stateIsOn}(\text{microwave}_0)$
$\{\text{stateOn}(\text{microwave}_0)\}$

Figure 8: GOALNET generalizes to unseen verbs such as *heat* when only the verb *boil* is seen at training.



Initial State
Instr.:*"fetch milk from the fridge"*

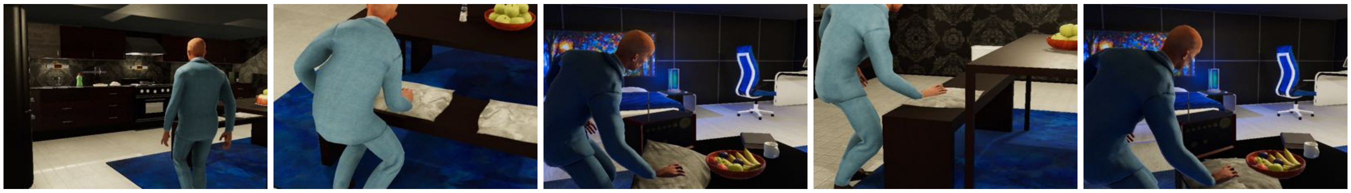
$\text{stateIsOpen}(\text{fridge}_0)$
$\{\text{MoveTo}(\text{robot}, \text{fridge}_0), \text{stateOpen}(\text{fridge}_0)\}$

$\text{ConnectedTo}(\text{milk}_0, \text{robot})$
$\{\text{MoveTo}(\text{robot}, \text{milk}_0), \text{Grasp}(\text{robot}, \text{milk}_0)\}$

$\text{OnTop}(\text{milk}_0, \text{counter}_0)$
$\{\text{MoveTo}(\text{robot}, \text{counter}_0), \text{PlaceOn}(\text{milk}_0, \text{counter}_0)\}$

$\text{stateIsClose}(\text{fridge}_0)$
$\{\text{MoveTo}(\text{robot}, \text{fridge}_0), \text{stateClose}(\text{fridge}_0)\}$

Figure 9: GOALNET generalizes to unseen verbs such as *fetch* when only the *get* and *bring* verbs are seen at training.



Initial State
Instr.: *"arrange pillows on the table"*

$\text{ConnectedTo}(\text{pillow}_0, \text{robot})$
$\{\text{MoveTo}(\text{robot}, \text{pillow}_0), \text{Grasp}(\text{robot}, \text{pillow}_0)\}$

$\text{OnTop}(\text{pillow}_0, \text{table}_0)$
$\{\text{MoveTo}(\text{robot}, \text{table}_0), \text{PlaceOn}(\text{pillow}_0, \text{table}_0)\}$

$\text{ConnectedTo}(\text{pillow}_1, \text{robot})$
$\{\text{MoveTo}(\text{robot}, \text{pillow}_1), \text{Grasp}(\text{robot}, \text{pillow}_1)\}$

$\text{OnTop}(\text{pillow}_1, \text{table}_0)$
$\{\text{MoveTo}(\text{robot}, \text{table}_0), \text{PlaceOn}(\text{pillow}_1, \text{table}_0)\}$

Figure 10: GOALNET can generalize even in case of paraphrased instructions.

tive constraints as relations $R_t^+(o_t^1, o_t^2)$ and $R_t^-(o_t^2, o_t^4)$. There are two identical decoder heads to independently predict the positive and negative constraints. To predict the relation in each constraint, a 1-layer FCN was used, with an output layer with size $\mathcal{S}$. We take the output of the Gumbel-softmax function and pass it to the decoder of the first object. The o_t^1 predictor generates a likelihood vector of size $\mathcal{O}(s_t)$ by passing it through a 1-layer FCN and the $\mathrm{softmax}$ activation function. The Gumbell-softmax output of the likelihood vectors of the relation and the first object are sent to the o_t^2 predictor to predict likelihoods for all object embeddings. This part was implemented as a 1-layer FCN with output size of $\mathcal{O}(s_t)$ followed by a $\mathrm{softmax}$ activation function.

Model Training. We use the Adam optimizer to train our model (Kingma and Ba 2014) with a learning rate of 5×10^{-4}. We use the early stopping criterion with loss on the validation set as the signal. We also decay the learning rate by $1/5$ every 50 epochs. In training, we use a constant teacher-forcing probability of $p = 0.2$ of using the planner to iteratively update the state instead of using ground-truth state-action sequence.

System Specifications. The GOALNET neural network is trained and evaluated on a machine with following hardware specifications: *CPU:* 2x Intel Xeon E5-2680 v3 2.5GHz/12-Core "Haswell", *GPU:* 2x NVIDIA K40 (12GB VRAM, 2880 CUDA cores), *Memory:* 16GB RAM.

C Additional Results

Performance with increasing complexity. Figure 7 characterizes the variation in SJI, IED, F1 and GRR scores as the size of ground truth constraint set ($(len(\delta^+) + len(\delta^-))$ increases. We observe graceful degradation in performance as constraint of model as the plan length increases. This suggests that there is a scope of improvement in the model to achieve good performance agnostic to plan length. However, we observe that the performance of GOALNET is better than the baseline for most cases. And this performance gap between the two models widens for larger constraint sets, showing that the neural approach in GOALNET is able to effectively encode the temporal context enabling it to outperform the baseline in multi-stage long-horizon tasks.

Generalization. Figures 8, 9, and 10 demonstrate the ability of GOALNET to generalize to language instructions unseen during training. These instructions correspond to verb frames that are *out-of-distribution* from the training data. Robust inference of conjunctive goal predicates enables the symbolic planner to generate feasible plans and reach goal states for these unseen tasks. Figure 8 shows a trace of the inferred goal-predicates and actions generated by the RINTANEN planner for an input language instruction of *"boiling the milk"* when the instruction uses the verb *heat* instead of *boil* in the training data. GOALNET correctly predicts the predicate of opening microwave, placing the bottle of milk inside and turning on the microwave to heat the milk. Similarly, Figure 9 shows a trace of predicates and actions in case the input language instruction is *"fetch milk from the fridge"* when the instruction has the unseen verb *fetch* instead of known verbs such as *get* or *bring*. Figure 10 shows another example where the model generalized to paraphrased sentences by performing the task correctly changing *"bring pillows to the table"* to *"arrange pillows on the table"*.