BusyBot: Learning to Interact, Reason, and Plan in a BusyBoard Environment (Supplementary Material)

1 Implementation Details

1.1 Interaction module.

Training detail. The interaction module is trained online in a self-supervised manner for 400 epochs. Each epoch contains a data collection phase and a training phase. During the data collection phase, 16 busyboard environments are generated and the agent executes 10 actions on each board. Each interaction data point is stored in a FIFO replay buffer (size=6400) as $\{o_t, a_t^{\text{pos}}, a_t^{\text{dir}}, r(a_t^{\text{dir}})\}$. During the training phase, in each iteration, we sample 16 and 32 interaction data points from the replay buffer for training position and direction inference network respectively. When sampling the data points, we make sure that half of the sampled data points have positive reward and half of them have zero reward. We train 8 iterations for the position network and 24 iterations for the direction network in each epoch.

Given the model structure, an effective direction depends on an effective position to gain the positive reward, thus to make the training more efficient, we only train the position inference network (with an initial learning rate of 0.0005 and Adam optimizer) and try all direction candidates in the first 100 epochs. If one of the direction candidates yields positive reward, the position network gains positive reward. To bootstrap the training, in the first 10 epochs, data are collected by a policy that chooses random positions and the training starts at the 10th epoch. From the 100th to 120th epoch, we randomly select direction candidates to collect negative data points. Starting from the 120th epoch, we jointly train the position and direction inference network (with an initial learning rate of 0.0001 and Adam optimizer). To encourage exploration, we apply the epsilon-greedy exploration algorithm with ε starting from 1 and decreasing linearly to a minimum value of 0.1 over a span of 40 epochs and 80 epochs, from the epochs when the training starts for the position and direction inference network respectively.

Network detail. The structure of the position inference network and direction inference network are discussed below:

Given a depth observation captured by a depth camera, we first calculate the world coordinates for each pixel using depth values. Surface normals are then estimated using the cross product of neighboring coordinates. Next, the depth image and surface normals are concatenated ($4 \times 480 \times 640$) and fed into the position inference network with a U-Net architecture. The position network has 4 down-sample blocks with 32, 64, 128, and 256 channels, followed by 4 up-sample blocks with 128, 64, 32, and 2 channels. Each down-sample (or up-sample) block includes a max-pooling (or bilinear interpolation) layer and two 3×3 convolution layers with ReLU activation. The output tensor has a size of $2 \times 480 \times 640$ and softmax activation is applied to obtain the final affordance score map ($1 \times 480 \times 640$).

The direction network takes in the current depth observation, surface normals, and the 2-D Gaussian representation of the selected position ($5 \times 480 \times 640$) and applies seven 3×3 convolution layers with 32, 64, 128, 256, 512, 512, and 512 channels. Max pooling is also applied except for the first layer. The output tensor with a size of $512 \times 7 \times 10$ is flattened as an embedding $\chi(o_t)$ and passed through a two-layer MLP with both 256 dimensions, followed by a four-layer MLP with dimensions of 1024, 1024, 1024, and 18. Finally, the network outputs the scores for all direction candidates (1×18).

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

1.2 Reasoning module.

Network details. The graph neural networks used in the reasoning module has a similar structure as the Interaction Network (IN) [1]. The frist GNN is a spatial encoder that takes in node $n_i(T \times N \times 259)$ and edge features $\{n_i, n_j\}(T \times N \times N \times 259)$, where T is the number of frames, N is the number of objects, and the 259-dimensional node feature is the concatenation of the object's 256-dimensional visual feature (extracted from the 10th layer of a pre-trained ResNet-50 network) and its 3D position. The input vectors are flattened and input to a sequence of linear layers followed by ReLU activation to encode (1 layer), propagate (4 layers for edges and 2 layers for nodes, in each propagation step), and decode (2 layers) the information. The objective is to learn f^{rel} and f^{obj} that maps the input features to node and edge embeddings,

$$h_{ij} = f^{rel}(n_i, n_j, \{e^d, e^h\}) \tag{1}$$

$$h_i = f^{obj}(n_i, \Sigma_{j \in N_i} h_{ij}) \tag{2}$$

where N_i denotes all objects that are directly connected by an edge to object i, h_i and h_{ij} are the learned spatial node ($T \times N \times 128$) and edge embeddings ($T \times N \times N \times 128$).

The other GNNs use the same structure as above to pass information through nodes and edges.

Training details. The total number of nodes N in the GNNs is set to 10, the maximum possible number of objects over all busyboards. For boards with less than 10 objects, we input 0's as placeholders for vacant nodes. In this way, the model is able to generalize to boards with different number of objects. In addition, since the model does not rely on the states of the triggers to make predictions, we find that the training can be more stable if we input 0's in place of visual features for trigger objects.

The action is encoded to a 256-dimensional embedding by a 3-layer MLP with 256, 256, 256 dimensions. For the object where the action is applied on, the input is the 6-D action; for other objects, the input are 0's.

We jointly train the relation inference and dynamics network for 200 epochs with a learning rate of 0.0005 and a batch size of 64, using the Adam optimizer. During training, we clip out sudden explosion in loss to obtain more stable training.

1.3 Planning module.

The relation agent compares the initial and goal image and identifies the different responders. For each responder that needs to be changed, the agent retrieves the corresponding trigger based on the functional scene graph (if multiple triggers are found, the agent randomly chooses one) and applies the action. Therefore, the total steps executed by the relation agent is the number of different responders in the initial and goal image and the relation agent will terminate regardless of whether the goal state is reached. The predictive agent and the BusyBot agent will terminate after executing 8 steps, or terminate immediately if the goal state is reached within 8 steps.

As for the learning-based agents (BC and PPO), at each step, both agents take the current observation o_t, the goal state observation o_g, the action candidates A_t, and the interaction history H_t as input, and infers a discrete action index for execution. For fair comparison, we make both the observations (top-down RGB images) and action candidates (inferred by the interaction module) the same as in our method. In addition, we encode the interaction history $H_t = [(o_0, a_o), (o_1, a_1) \cdots, (o_{t-1}, a_{t-1})]$ via a LSTM as an implicit scene representation for each step. The objective of PPO is to optimize the cumulative reward (i.e., success rate) and the objective for BC is to mimic the behavior of an expert agent (generated using an oracle policy).

2 Environment Details

Board Generation. The main body of the board will be assigned random colors (from 8 colors) and textures (from 5 textures) upon generation to introduce variety. Objects are placed into random positions on the board with no overlap.

Objects from the lamp and door category may rotate $\theta_z = \{0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi\}$, where θ_z denotes rotation (in radian) along the z-axis counter-clockwise. The orientation of objects from the switch and tracktoy

category are fixed in order to eliminate possible ambiguities. The joint states of the triggers are also randomly set upon board generation to add variance to initial board observations.

Relation Assignment. Switch objects are classified into small-displacement, multi-direction, and multi-link triggers. Door objects are single-stage motion effects. Tracktoy objects are multi-stage motion effects. Lamp objects can be either single-stage or multi-stage appearance effects. When sampling the relation, we make sure that each trigger object is matched with at least one responder.

3 Result Details

Figure 1 and 2 shows the step-by-step reasoning result on simulated and real-world busyboards. We highlight the object that the agent interacts with in each step using the yellow symbol. We can observe that the edge predictions associated with each object refine through interactions.

Figure 3 shows the step-by-step goal-conditioned planning results on boards with novel configurations. All agents perform well on one-to-one tasks, while the predictive agent and BusyBot agent outperform the relation agent on one-to-many tasks by leveraging future predictions to select the correct link or correct direction.

Figure 4 shows step-by-step goal-conditioned planning result on boards with novel objects. One-to-one (a) shows a case when all agents perform well, demonstrating that the learned relation and dynamics can generalize to boards with unseen object instances. One-to-one (b) shows a case when incorrect future predictions lead to a repeating action (the agent turns on and off the green light), while the relation agent and BusyBot agent are able to retrieve the correct trigger to manipulate using the functional scene graph.

The one-to-many task on boards with novel objects is challenging for both relation and predictive agent, given that the relation agent cannot infer the exact action position or direction, and the predictive agent might not be able to select the correct action either as the future state prediction accuracy is low on novel objects. By combining the advantages of both agent, the BusyBot agent may yield the correct result by narrowing down possible action candidates using the functional scene graph and leveraging future predictions of the dynamics network to choose the correct action.

References

[1] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

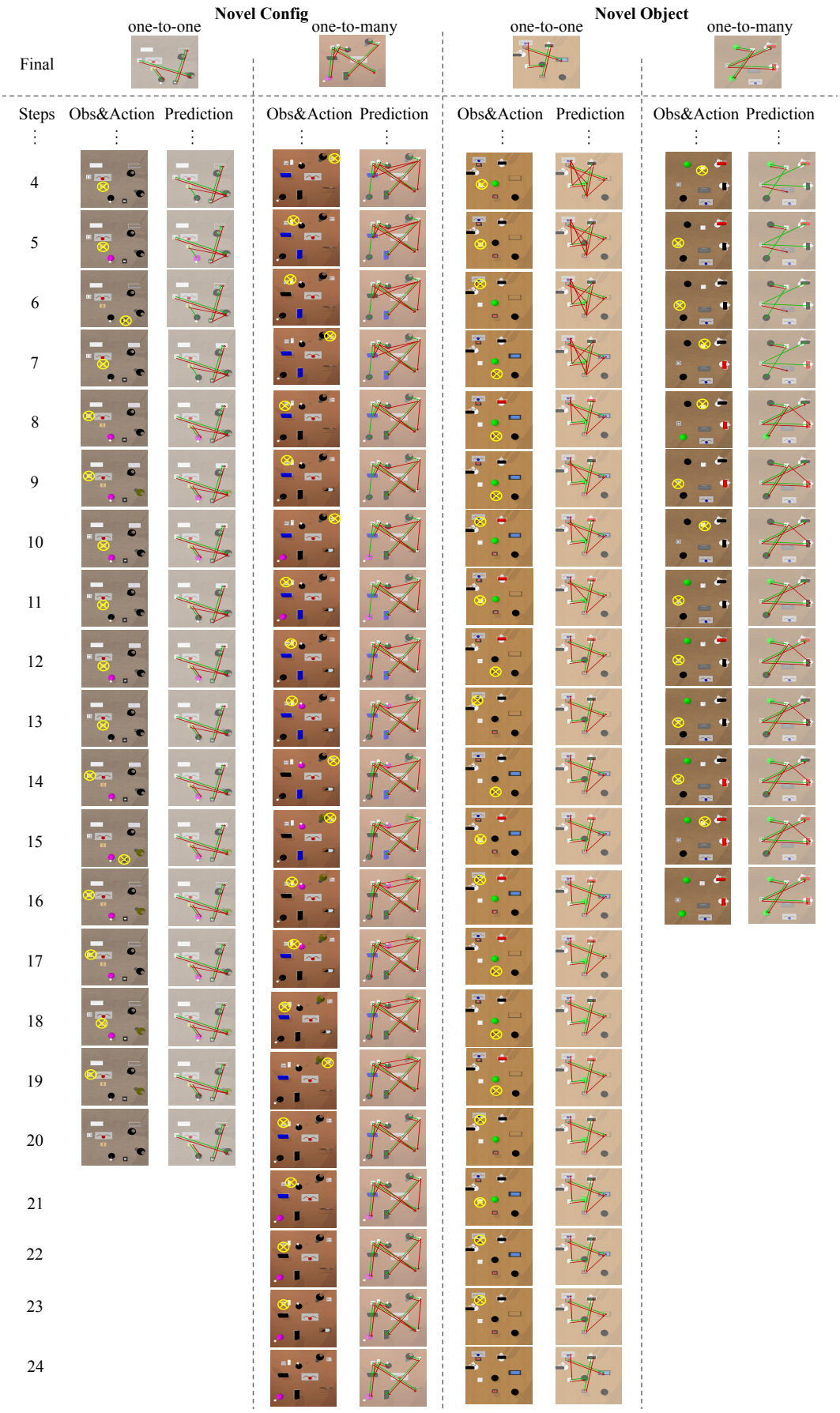

Figure 1: **Full reasoning sequences on simulated busyboards.**

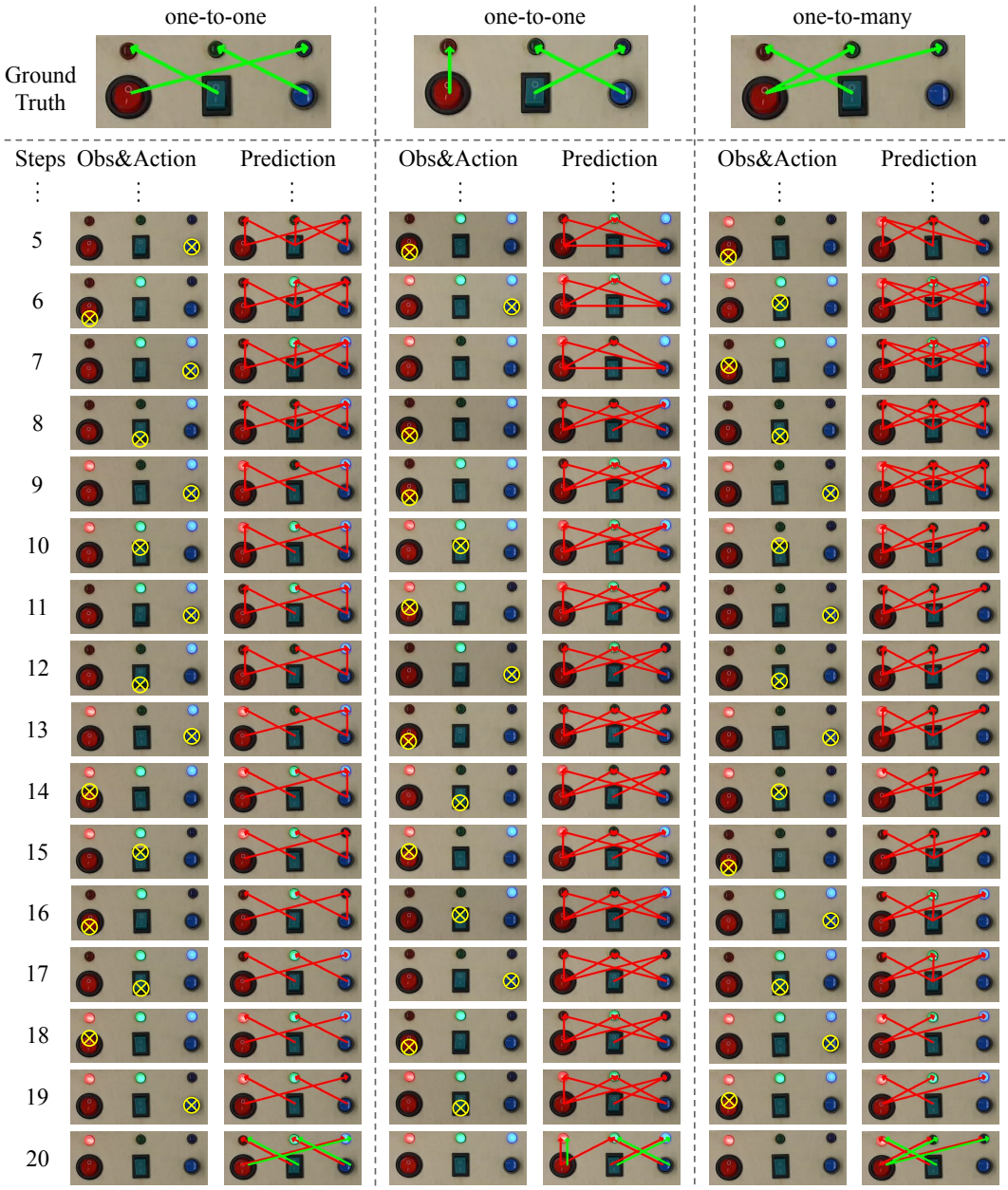

Figure 2: **Full reasoning sequences on real-world busyboard.**

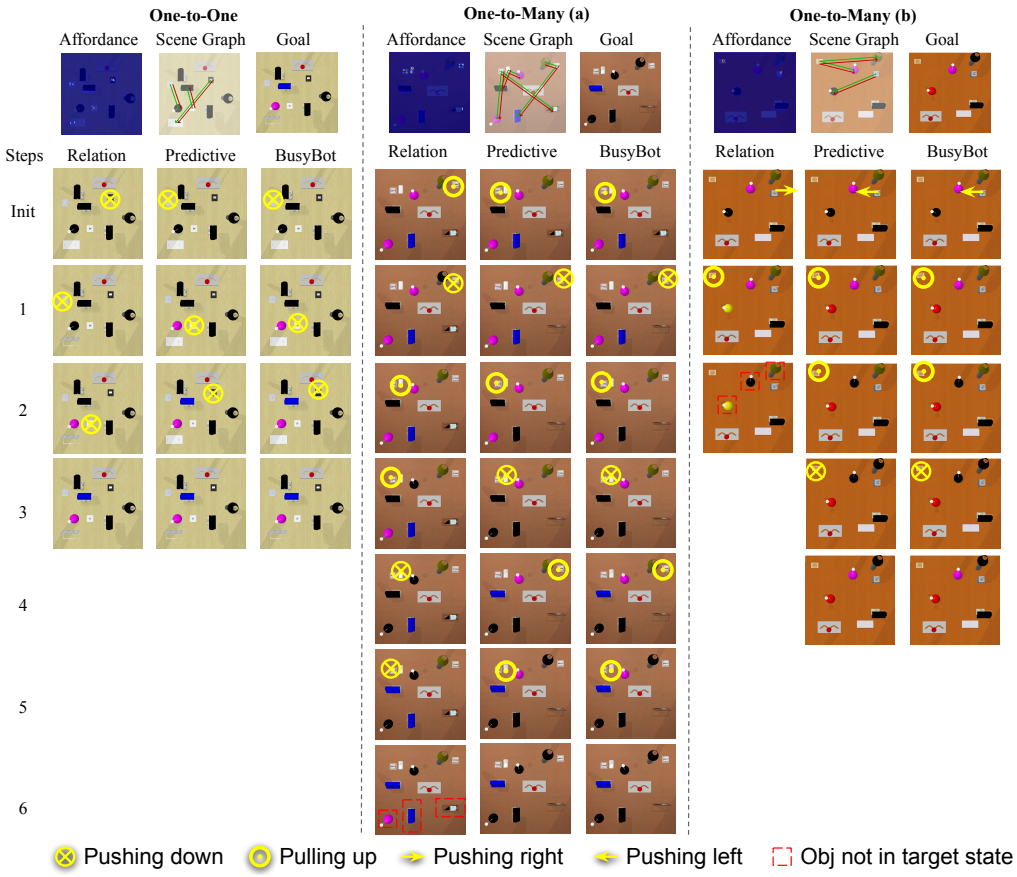

Figure 3: **Goal-conditioned planning result on boards with novel configurations.**

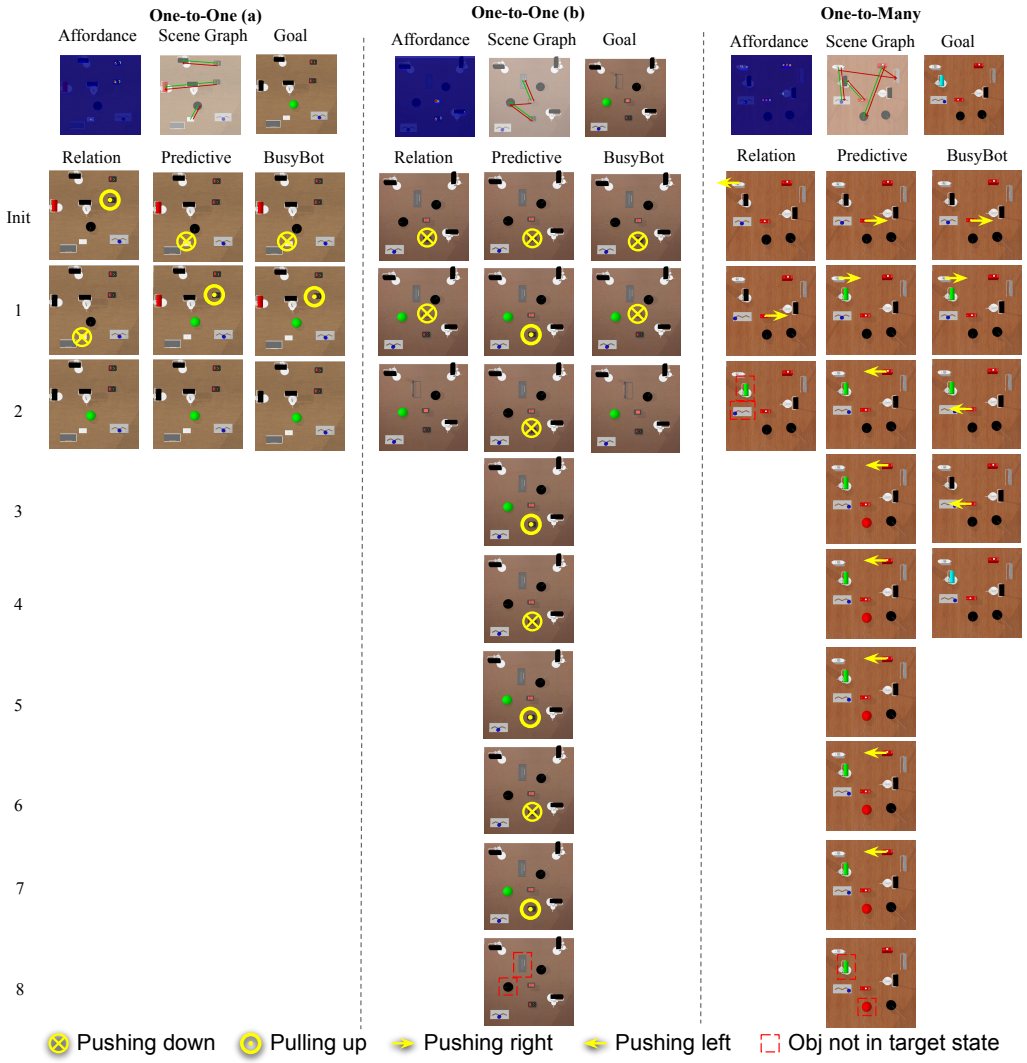

Figure 4: **Goal-conditioned planning result on boards with novel objects.**