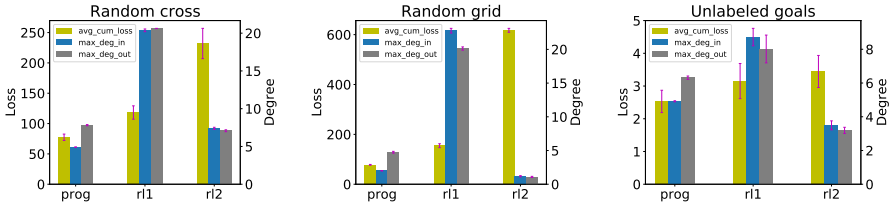
We thank the reviewers for their comments and will do our best to address them in our paper.

R1, comparison to RL with communications as actions. We agree that the multi-agent communication problem can be formulated as an MDP where decisions about which agents to communicate with is part of the action. We performed additional experiments to compare to this approach. As seen in the plots below, this approach performs poorly compared to ours. We believe this is because of the combinatorial blowup in the action space—i.e., there is a binary communication decision for each pair of agents, so the number of communication actions is $2^{N(N-1)}$ (where N is the number of agents). Since the action space now includes discrete actions, we use the policy gradient algorithm to train the policy. We tuned several hyperparameters including (i) weights for balancing the reward term with the communication cost, (ii) whether to use a shaped reward function, and (iii) whether to initialize the policy with the pretrained transformer policy. In the figure, rl1 corresponds to the baseline policy that achieves the lowest loss across all hyperparameters we tried; however this policy has a very high communication degree. In addition, rl2 corresponds to the policy with lowest communication degree, but this policy has very high loss. In summary, learning the communication policy directly using RL is challenging due to the large discrete action space, and our proposed approach addresses this challenge by using the transformer as a teacher.



R1, "manually picked" rules. The communication rules used to solve each benchmark are **not chosen manually**. Instead, they are automatically learned by the search algorithm (based on MCMC) in Section 3.4. The search space is encoded by the domain-specific language (DSL) in Section 3.2. In particular, this search space consists of programs composed of arbitrary combinations of deterministic rules (with the argmax operator) and stochastic rules (with the random operator). In addition, the search algorithm must determine (i) what predicate B to use in each filter operator, (ii) what function F to use in each map operator, and (iii) the weights β in each predicate/function. Finally, programs can contain more than two rules. The number of rules is a hyperparameter; we choose this parameter using cross-validation.

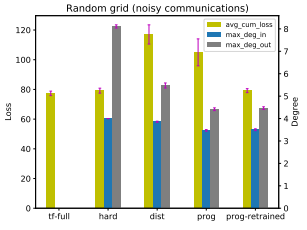
R1: program synthesis terminology. We uses the term "program" since the rules rely on map/filter operators that cannot be captured by if-then-else rules. Also, our DSL is similar to the ones used in prior work—e.g.:

Verma, A. et al. Programmatically interpretable reinforcement learning. ICML, 2018.

R1, Figure 2. In Figure 2c in our paper, the loss is computed as $(10 - \text{reward})$. We will clarify this in our paper.

R3, full communication. In our approach, the number of rules in the program is a hyperparameter K. Given K, our algorithm learns a program that optimizes performance while communicating with at most K other agents (in terms of in-degree). In our experiments, the range for this hyperparameter is $K \in \{2, 3, 4, 5\}$. When we set this hyperparameter to be large, we empirically observe that each agent communicates with many other agents, so we expect that it would learn the full-communication policy as long as K is sufficiently large. Our algorithm uses cross-validation to automatically choose an appropriate K in the given range of possibilities.

R3, noisy communication. We have added a new benchmark based on the existing random grid task, but where the communications between any two pairs of agents has a 50% probability of failing. The results are shown in the adjacent figure. As can be seen, the programmatic communication policy has similar loss as the transformer policy while simultaneously achieving lower communication degree. Here, the best performing policy has four rules (i.e., hyperparameter $K = 4$), whereas for the existing random grid task, the programmatic policy has 2 rules. Intuitively, agents are attempting to communicate with more of the other agents to compensate for lost communications.



R3, non-navigation domains. We note that the unlabeled goals task in the paper is not just a navigation task. In this task, the agents are not pre-assigned goals, so there is a combinatorial aspect where they must communicate to assign themselves to different goals. The main difference between this task and capture the flag is that the goals are static, whereas they are moving (typically adversarially) in capture the flag. We believe our approach can be applied even in adversarial settings, and will do our best to add an additional task along these lines to our paper.