We really appreciate all reviewers for their careful reviews and constructive comments. We are pleased that all reviewers find our work innovative and interesting. In the following, we highlight the primary issues and questions proposed by the reviewers. Remaining minor comments and suggestions will also be incorporated in the revision to polish our paper.

Primary Issues

The importance of curriculum learning (R1, R4). Thank R1 and R4 for pointing it out. As supposed by R4, the main reason why curriculum learning seems to be so important is that our model uses reinforcement learning. It is not easy to make the model practically effective since its action space is exponentially large, which is due to the indefinite length of output sequences in Solver, and the huge number of possible trees in Composer. Such a huge space means that rewards are very sparse, especially for harder examples. Without curriculum learning, our randomly initialized model receives zero rewards on most examples and shows no sign of convergence even after training for several days. Therefore, test (even train) accuracies in Table 3 are zero across all tasks. In comparison, by arranging examples from easy to hard, curriculum learning alleviates the sparse reward issue. On the one hand, easy examples are more likely to provide non-zero rewards to help our model converge; on the other hand, models trained on easy examples have a greater possibility to receive non-zero rewards on hard examples. Regarding R1's consideration about different curriculum settings, we have not tried others since we find the simplest length-based curriculum effective enough. We think other settings such as MCD-based curriculum may also work. We will include the above explanation in the revision.

The clarification of "extra resources" (R1, R3). We clarify "extra resources" here as "data or data specific rules other than original training data". GECA and Meta Seq2Seq lie in "extra resources" since they both utilize extra data. Specifically, GECA recombines real examples to construct extra data, while Meta Seq2Seq employs random assignment of the primitive instructions (e.g. "jump") to their meaning (e.g. `JUMP`) to synthesize extra data. Regarding data-specific rules, Equivariant Seq2Seq requires predefined local groups to make the model aware of equivariance between verbs or directions (line 321, e.g. "jump" and "run" are verbs). Similarly, Program Synthesis needs a predefined meta grammar, which heavily relies on the grammar of a dataset, and hence we think it also falls into the group of using "extra resources". In comparison, other methods including ours (Table 1) do not utilize any synthetic data or data specific rules. We will define "extra resources" explicitly in the revision.

The representation of methodology section (R1, R2, R3). Thanks for all the kind suggestions, and we will carefully proofread the section to achieve better readability in the revision. First, we will give a clear picture of how the overall system works at the beginning of Section 3, and explicitly mark the neural parts and symbolic parts. Second, we will utilize two figures instead of Figure 2 to illustrate the overall system and the detail of Composer respectively. In order to make the figures easily understood, we will polish all the captions with enough information. Last, we will invite native speakers to help us thoroughly proofread our paper for grammar and style.

Other Questions

Explain the behavior on "$x after $y twice" (R1). In fact, our model does not depend on a specific order when finding the recognizable SrcExp. The different behaviors in R1's cases are normal since the finding process is context sensitive. Taking "$x after $y twice" as an example, our model will assign the highest merging score on "$y twice", which means that it will first merge "$y twice‘' in the context of "$x after $y twice".

What kind of grammar does the model learn (R2). We find that the grammar learned by our model is very similar to the phrase-structure grammar defined in SACN. Regarding the difference, because of our designed simplicity-based reward, our model tends to learn rules such as "D -> X Y", "Y -> left" and "Y -> right", instead of "D -> U left" and "D -> U right" in the SCAN grammar. These two kinds of rules are both valid on SCAN. And we will provide more examples for better understanding the learnings of our model.

More analysis on MiniSCAN and real-world settings (R3). We also think compositional generalization on few-shot tasks and real-world applications are interesting. We will follow R3's advice to complement more analysis on our MiniSCAN experiments. As for real-world applications, one of our future work is to experiment on the dataset CFQ.

Extend this method to CFQ (R4). Although still under progress, we believe that with some modifications on Solver (i.e., switching from "NL to actions" to "NL to SPARQL"), our method can readily generalize to the CFQ dataset.

Details about training devices and time (R4). Our model was trained on a single Tesla-P100 (16GB) and the training time for a single run is about 20 $\sim$ 25 hours. We will add it to our paper in the revision. For each result of our method, we run 5 replications, as indicated in the caption of Table 1.

What kind of human prior knowledge is learned (R4). The prior knowledge here is referred to the phrase-structure grammar defined in SCAN. This claim is based on the observation that our model can achieve perfect performance without injecting such knowledge manually, indicating that our model learns this knowledge to some extent.