We would like to thank all three reviewers for their thoughtful comments. We are pleased with the generally positive reception of our work, and we will make sure to incorporate all of the helpful feedback into the camera-ready version.

**Relation to prior synthesis approaches.** R3 saw our approach as "very similar to the standard approach for neural program synthesis from input-output examples, which typically assumes that the number of input-output examples is small, e.g., no more than 10." We believe our model actually differs significantly from previous approaches in this regard. Previous neural I/O synthesis models, such as RobustFill, as well as [17], [18], [10]—designed for a small, fixed number of examples—generally use a *separate* encoder-decoder model (possibly with attention) for *each* example. Information from the separate examples is only combined through a max-pool or vector concatenation bottleneck—there is no attention *across* examples. This makes these models unsuitable for domains where it is necessary to integrate relevant information across a large number of diverse examples. To confirm this, we implemented RobustFill on our MiniSCAN domain, which only achieves 3%, 4%, 3%, 3.5% accuracy on 3-6 higher-order rules, respectively. In contrast, our model encodes each I/O example with an example encoder (ln 115), and then a single decoder model (ln 119) attends over these example vectors while decoding. By attending *across* examples, our approach can focus on the relevant examples at each decoding step. This is particularly important for our domains because there are many examples, and only a subset are relevant for each decoding step (i.e., each rule). We agree that we did not highlight this distinction enough in our submission; we thank R3 for pointing this out, and will incorporate this discussion and new RobustFill results into our paper. R1 notes that our submitted code seems to implement a more complicated network than is described in our submission. While our code is able to perform a "double attention" mechanism, this work does not use these features of the code, and the architecture is as described in the paper. We thank R1 and apologize for this confusion.

**Suitability of our approach to SCAN.** According to R2, our paper "shows quite convincingly that neural program synthesis methods can infer grammars like the one used in SCAN from a relatively small amount of examples." On the other hand, R3 questions if our approach is suitable for SCAN because it only considers up to 100 examples at a time, whereas the SCAN training sets contain >10,000 examples. We agree with R2; we see this as an advantage of our system rather than a weakness. While previous approaches require training on tens of thousands of examples, our results demonstrate that using our method, 100 examples are enough to achieve perfect performance on 4 SCAN splits.

**Support set selection.** R1 and R3 also note that we use heuristics to select support set examples for SCAN. As discussed in the supplement (ln 50-61), our heuristics are guided by general principles: we seek to match the distribution of examples in the train and test sets. The SCAN dataset is formed by *enumerating* all possible examples from the SCAN grammar up to a fixed depth; our models were trained by *sampling* examples from the target grammar. This causes a distributional mismatch which we rectify by upsampling shorter examples at test time, while ensuring that all rules are demonstrated. We ran an additional experiment, re-generating the SCAN data by sampling instead of enumerating. If we condition our model on support examples *sampled* from the underlying SCAN grammar, we are also able to solve SCAN from 100 examples. Our revision will report this experiment and move the discussion on the heuristics to the main text.

**Search.** Our approach utilizes test-time search, which R3 also suggests is a disadvantage: "The results of [the no search baseline] somehow further suggest that the model does not really learn the grammar rules correctly." We respectfully disagree. We see the entire neuro-symbolic system—rather than just the neural component—as the model; in particular, the search procedure is an asset of our method. Indeed, our results demonstrate how, because we use test-time search, our neural network does not need to be perfectly trained for our overall model to yield excellent prediction accuracy. We contrast this with neural-only approaches (as well as the no-search baseline) which require very careful training in order to achieve perfect results. In that sense, our approach offers more robustness than a neural-only model would allow.

**Meta-grammar.** The reviewers note that our model uses strong supervision in the form of a meta-grammar. We have shown how training on examples from a family of grammars can lead to very high accuracy. Although used in this work, this amount of strong supervision may not be needed. We see our work as an important step in a larger line of work, which can extend our approach to: incorporate grammars which support noise (R2), incorporate RL to replace much of the strong supervision, and learn the execution model. In a sense, we agree with R2: "Now that this paper has shown how to solve SCAN, maybe the right answer for the community is to move towards harder and more realistic datasets." R3 suggests extending our MiniSCAN experiment to test on grammars with many more rules than the 2-4 in the training meta-grammar. We report % accuracy here, for testing on 7-13 higher-order rules, respectively: synth: 96.0, 93.6, 92.0, 90.5, 83.5, 78.5, 77.5; no search: 59.0, 62.0, 62.5, 56.0, 59.5, 48.5, 52.5; meta Seq2Seq: 58.5, 59.8, 69.0, 62.5, 56.5, 55.5, 53.5. This demonstrates both generalization and graceful degradation on grammars with 3x the number of rules vs training.

**Additional questions.** R3 asks if the Chinese and Japanese number systems are the same; they are not. For example, they differ when the leading digit is 1, or when 0 is a middle digit. R2 asks how tokens are encoded. See supp. ln 93-97 for details on how we can tokenize in a symbol-agnostic manner. R1 asks how we ensured that grammars are properly formatted. If the model predicts a sequence which does not parse into a valid grammar, it is simply discarded and another is sampled. R1 asks about errors in the number word experiments. French, for example, contains irregularities such as 80 $\rightarrow$ quatre-vingts ("four twenty"), which our grammar did not support. We will discuss this in camera-ready.