We would like to thank the reviewers for their feedback. We will add the suggested references, clarifications from our answers below, and further qualitative experiments in the camera-ready revision.

[R1] Which parts of the model in §4 constitute the baseline? The AST-based decoder described in §4.1 constitutes our baseline. Its architecture largely follows Yin and Neubig [3]. PATOIS adds two novel contributions on top of it, described in §4.2: **(a)** the new objective in Eq. (4) that allows the model to emit idiom rules instead of original CFG rules, and **(b)** the new training regime that teacher-forces idiom bodies when an idiom rule is chosen. We will specify this more clearly in the camera-ready revision.

[R1] How does PATOIS compare against methods based on sketch generation? Coarse2Fine and Bayou both require a manually-defined formulation of program sketches unlike PATOIS which learns idioms from data. There only exists one sketch per program, unlike idioms which can occur at multiple places within the program. Fig. 1 shows that when run on the Hearthstone test set, PATOIS invokes 4–15 idioms in the course of decoding. Moreover, larger programs use more idioms, showing that the decoder of PATOIS learns to switch between high-level and low-level reasoning repeatedly, rather than learning only high-level sketches.

[R1] What are the possible failure modes of idiom mining? The most important failure mode of idiom mining is proposing idioms in $\widetilde{\mathcal{I}}$ that end up unused by the synthesis model despite being common. For instance, our best Hearthstone model never used 29 out of $K = 80$ idioms on the test set despite them matching in some ASTs. Another possible failure is overfitting idioms to the training set. We tackle it by filtering out idioms that do not occur in the validation set, and empirically demonstrate that the remaining ones generalize to the test set. In general, we evaluate $\widetilde{\mathcal{I}}$ as a hyperparameter, choosing the vocabulary that optimizes validation performance (as in Tables 2-3). We have ongoing work on an end-to-end extension of PATOIS, but this extension is beyond the scope of this work.

[R2] What is the runtime complexity? PATOIS has two phases: (1) idiom mining and (2) training the synthesis model. We outline the complexity analysis for them below, and will add a complete one to the camera-ready revision.

Phase 1 implements MCMC sampling, run for $M = 10$ iterations. At each iteration, PATOIS traverses each AST $T \in \mathcal{D}$ once to sample the random variables that partition it into the idiom fragments (see §3). Thus, the complexity of mining is $\mathcal{O}(M \cdot \sum_{T \in \mathcal{D}} |T|)$. In practice, for the 10,181 ASTs in Spider it took < 30 min on a 32-core 2.4 GHz Intel Xeon®.

Phase 2 has essentially the same complexity as the baseline training thanks to our objective in Eq. (4). For a training instance $\langle X, T \rangle$ computing the loss takes $\mathcal{O}(|T|)$ time. Each step computes cross-entropy between the predicted distribution over production rules and the one-hot distribution with the ground truth rule. In Eq. (4), the cross-entropy now allows > 1 ground truth rules: the original CFG rule and any matching idioms. The asymptotic complexity of cross-entropy is the same, so the overall per-instance complexity remains at $\mathcal{O}(|T|)$; however, there are more production rules involved, which increases the cost of computing the predicted distribution itself.

[R1, R3] Is the method applicable to state-of-the-art models? What would be the improvement? At the time of writing, the state of the art on Hearthstone and Spider is achieved by GrammarCNN [2] and IRNet [1], respectively. Notably, both of them (like many other contemporary models) use *structural AST-based decoders*, trained using the cross-entropy objective over the AST production rules. As we describe in §4, the PATOIS framework is applicable to any decoder that follows such architecture. We only compared against our baseline for fairness, but may be able to also implement PATOIS on top of the open-sourced GrammarCNN for the camera-ready revision.

The improvement of PATOIS should benefit any such structural decoder because idioms fundamentally can help to *avoid mistakes in modeling the generation of idiom bodies* (which account for a sizable fraction of the AST). The effect will be less prominent for IRNet where the Coarse2Fine sketching mechanism partially accomplishes the same goal, and more prominent for GrammarCNN.

[R2, R3] Please provide more qualitative experiments. How often are the idioms used? We agree that §5 needs more qualitative experiments, and will add them to the camera-ready revision. We conducted some during the author response period to supplement our answers. For instance, Fig. 1 shows a distribution of idiom usage on the Hearthstone test set.

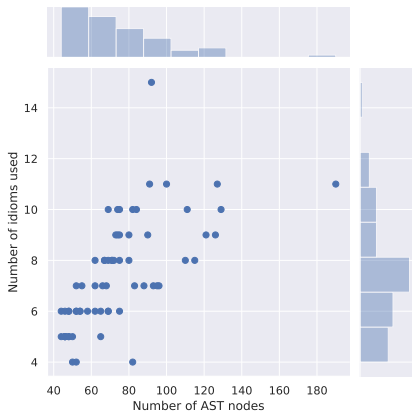

Figure 1: Hearthstone idiom usage on the test set. Number of idioms used and AST nodes are from synthesized trees, not ground truth.

[1] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu, and D. Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *ACL*, July 2019.

[2] Z. Sun, Q. Zhu, L. Mou, Y. Xiong, G. Li, and L. Zhang. A grammar-based structural CNN decoder for code generation. In *AAAI*, 2019.

[3] P. Yin and G. Neubig. A syntactic neural model for general-purpose code generation. In *ACL*, July 2017.