We thank the reviewers for their thoughtful comments. We are delighted that reviewers unanimously find our work novel, well-written, and state-of-the-art on well-studied benchmarks. Most of their questions ask for additional analysis and ablation experiments. Below we try to provide as many of them as we could accomplish within the tight time frame.

R1, R3, R4: Where does the performance gain come from—the transition system or the GNN? We perform an ablation study that only changes the transition system while keeping GNNs untouched. We implement In-order Shift-reduce System (ISR) [22]. In order to apply GNNs to ISR, we rely on Theorem 3 to interpret ISR states (stacks) as augmented partial trees. ISR + GNNs (with XLNet encoder) achieves an F1 score of 96.24 on PTB, which is lower than our method (96.34 ± 0.03). This ablation demonstrates that the attach-juxtapose transition system contributes to the performance.

R4: Does the choice of GNNs matter? Besides GNNs, what about other tree encoders? We experimented with TreeLSTMs [39] and multiple GNN architectures such as GATs [Veličković et al. ICLR 2018]. Some of them were trained faster than GCNs, but they converged to a similar final performance. We prefer GNNs to TreeLSTMs because they work for graphs with loops. Although our graph is a tree without loops, variants of our method could violate the tree constraint by introducing additional edges,e.g., between consecutive tokens. We chose GCNs among other GNNs because it is straightforward to separate content and position information (details in the supplementary material).

R1, R4: Detailed analysis of the performance, perhaps using Berkeley Parser Analyser. We use Berkeley Parser Analyzer [Kummerfeld et al. EMNLP 2012] to categorize the errors of Mrini et al. [27] and our model on PTB. Two methods have the same relative ordering of error categories. The 3 most frequent categories are "PP Attachment", "Single Word Phrase", and "Unary". Compared to Mrini et al., our method has more "PP Attachment" (342 vs. 320) and "UNSET move" (33 vs. 23), but fewer "Clause Attachment" (110 vs. 122) and "XoverX Unary" (48 vs. 56). We will present the detailed results in the revised paper.

R1, R3: Compare with existing parsers in terms of efficiency. Our method is slightly faster than existing parsers, measured by the wall time for parsing the 2,416 PTB testing examples. It takes 33.9 ± 0.3 seconds for our method (with XLNet, without beam search), 37.3 ± 0.2 seconds for Zhou and Zhao [49], and 40.8 ± 0.9 seconds for Mrini et al. [27]. About 50% of the time is spent on the XLNet encoder, which is the same computation for all three methods. We run these experiments on machines with 2 CPU cores, 16GB memory, and one Nvidia GeForce GTX 2080 Ti GPU.

R3: Additional baselines. We experiment with the two baselines suggested by R3. As the first baseline, we compute attention for the rightmost chain using token embeddings at each node's starting position. It achieves an F1 score of 96.18 on PTB, which is between the sequence-based ablation (95.54 ± 0.07) and our method (96.34 ± 0.03). As the second baseline, we compute attentions for `attach` and `juxtapose` actions separately. It achieves an F1 score of 96.35 on PTB. It is not clear whether this is better or worse than our original method (96.35 vs. 96.34 ± 0.03). However, we will investigate more closely and include the results in the revised paper.

R3: Limited novelty compared to previous incremental parsers. The contribution is a more expressive parameterization of parsing action prediction. We respectfully disagree that our contribution is "a more expressive parameterization." Compared to the closest incremental parser (Collins and Roark [6]), our main novelty is in the action space itself, rather than how it is parameterized. Our actions can produce any valid tree (Theorem 1), whereas Collins and Roark can produce only a subset of them. This is because they rely on grammar rules and additional rules prescribing what structures are allowed in parse trees (They call them "allowable chains" and "allowable triples"). These rules are necessary for making their search space manageable, but they make it impossible to produce some trees.

R3: How is the attention computed in the sequence-based ablation? No attention is computed. At each step, we simply predict the target node as an integer in $[0, 249]$. It works because the rightmost chain is shorter than (or equal to) the sentence length, and all sentences in the datasets are shorter than 250.

R4: Why does beam search help so little? We were also surprised to find that beam search helps only marginally. A possible reason is that our method is trained with a local loss at each step, whereas prior work has demonstrated beam search works most effectively when combined with global losses. [1]

R1, R2: The potential impact of incremental parsing on NLP. Besides psycholinguistic motivation, incremental parsing is also useful in NLP applications. It produces a parse tree before a complete sentence is available, which is desirable when the agent responds to streaming input in real-time. For example, [2] a human-like conversational agent needs to process input information incrementally, since humans do not wait until the end of every sentence to respond.

R2: Evaluate the method on speeches rather than texts. This is a good idea since speech is a domain where it is more important to parse incrementally. However, that is out of the scope of this paper. Also, most existing incremental parsers [6,7,31] were evaluated on texts.

[1]Zhang and Nivre. "Analyzing the Effect of Global Learning and Beam-search on Transition-based Dependency Parsing", COLING 2012

[2]Schlangen and Skantze, "A General, Abstract Model of Incremental Dialogue Processing", EACL 2009