We thank all four reviewers for their thoughtful reviews, and are happy that they value the contribution of a new expressive, unifying formalism that maintains tractable inference. We acknowledge that this is a purely theoretical paper; an implementation is on our agenda, but we felt that there was already more than enough here to write up. Reviewer 4 (**R4**) asks for "at least an example of a problem in which it would be more useful than other frameworks." We offer, if we may, a recent exchange on Twitter,[1] where Charles Sutton asked whether there was a procedure that could go from a description of a PCFG to an efficient inference algorithm for trees given strings. In our opinion, no satisfactory answer was given, but FGGs provide such a procedure, sketched in Appendix A. We assumed that the CFG is in Chomsky normal form, but Lemma 15 would automate that as well.

Formalism **R2** gives an example FGG that appears to generate graphs with arbitrarily high treewidth. If we understand the notation correctly, Rule R2 has two hyperedges on its left-hand side, which is not allowed by the definition of HRG.

R4 provides feedback about our notation for left-hand sides, which are drawn as a hyperedge together with the same number of external nodes as the right-hand side, to show the pattern that gets replaced with the right-hand side; sorry if this wasn't clear. We are uncertain how **R4**'s suggestion to draw the external nodes on the left-hand side differs from what we already do. (Except Figure 2 – thanks for drawing our attention to the missing external nodes there.)

Renaming We agree with **R3** and **R4** that Def. 7 should formalize how nodes are renamed when copied. We'll fix it, but we'd also like to clarify two points. First, **R3** writes that we want to know which nodes have the same name so we can query by name. But this wouldn't be as flexible as we'd like; for example, we'd like to query a HMM for the probability that the last tag is N, but the last tag has different names in different graphs. Enabling such queries is why we developed conjunction. Second, **R4** writes that we need node names when matching up nodes during conjunction. But conjunction is an operation on FGGs, not factor graphs, so at the time of conjunction, no renaming has taken place.

Ambiguity **R3** asks whether there is spurious ambiguity due to the ordering of rewrites. We are assuming (and can make explicit) that a derivation is a tree, not a sequence of rewrites, so we don't count different orderings separately in the sum-product. However, **R4** asks about different derivation *trees* yielding the same graph, and we do treat these separately in the disjoint union and count them separately in the sum-product, so indeed, a FGG defines a distribution over (derivation, assignment) pairs. We agree that the notation should be improved and will think about how to do so.

Exact inference **R2** and **R4** ask about the claim (line 190) that sum-product is linear in the size of the graph. It's linear in graph size and exponential in the treewidth k, but k is a property of the grammar, independent of graph size.

R4 asks about the claim on line 192 that Lemma 15 reduces treewidth; we apologize that this is indeed not worded correctly. Lemma 15 does not change the generated graphs and cannot change their treewidth. Rather, Theorem 16's time complexity depends on the size of the right-hand sides, and Lemma 15 reduces the size of the right-hand sides.

R4 asks why case (1) and/or (2) at lines 223–5 are not applicable if the condition at line 230 is not satisfied. If X can derive a graph with two occurrences of X, then we can't solve the resulting system of equations by substitution because of the circularity, and we can't solve it using linear algebra because some of the equations are quadratic.

Approximate inference **R1** asks whether approximate inference can be done on FGGs. This was one motivation behind §5.2, since converting a finite FGG to a factor graph makes it possible to use any inference algorithm. Approximate inference on infinite FGGs remains an open question, which we consider an important direction for future research.

RBMs **R1** and **R3** ask to clarify our claim that FGGs can't generate RBMs. We mean that an FGG can't generate the infinite set of all RBMs; this example comes from Obermeyer et al. (2019) and would be of interest in a situation where one wants an FGG that can be conjoined with another to yield a single RBM. Since an $m \times n$ RBM has treewidth $\min(m, n)$, the set of all RBMs has unbounded treewidth and can't be generated by a FGG.

Other formalisms **R1** asks how the complexity of inference for FGGs compares to the complexity of inference in the formalisms we compare them to. We think the complexity should be the same in the cases considered in Appendix B and can expand the proofs to include complexity analyses.

R2 asks for comparison with PRMs, **R4** for comparison with PPLs, and **R3** for both. Our comparisons were focused on more constrained formalisms, but we'll be happy to discuss these more general formalisms as space permits. We suspect both PRMs and PPLs can simulate any FGG, but doubt whether the simulation would allow exact inference with the same efficiency. **R2** mentions tractable relational models, and we'd be interested in looking further into any citations that **R2** might provide.

Thanks We thank the reviewers for providing other missing citations, especially **R3**, and we are grateful for the many other comments from all the reviewers, which we will address in the final version of the paper.

[1] `https://twitter.com/RandomlyWalking/status/1258307220371984384`