Report on *Identities of the Kauffman monoid* $\mathcal{K}_3$.

This lovely article presents a description of when an identity holds on the Kauffman monoid $\mathcal{K}_3$: a combinatorial characterisation, and then a polynomial time algorithm. The article also gives some interesting discussion of the identities of other related semigroups and monoids, including a little on Kauffman monoids $\mathcal{K}_n$, with $n > 3$.

The article is a delight to read, and the results are interesting and nontrivial. It's very easy to recommend publication of the article, with minimal changes suggested.

Corrections. If only more articles were as cleanly presented as this! Despite reading carefully, I have only a few comments.

page 2, 8th last line of the Introduction. This sentence reads slightly strangely. Perhaps "With the inclusion of Kitov and Volkov on the team, we have mastered..."

Page 9, line -12. words $\mapsto$ word.

Page 10, line 24 (or -20). This comment can probably be ignored, but it was something that initially confused me, so I'll mention it. When you write "remove **the** occurrence" the reader is being expected to know that there is only ever one occurrence of any letter; but at this point it isn't clear. I concede that you are trying to construct $\overleftarrow{v}$, which should have only one occurrence of each letter, but during its construction it is possible it could have more. Another inconsequential comment here: I would have written "initialize $\overleftarrow{v}$ **as** the empty word". Finally, a trivial typo: I think there is an extra space at the end of the sentence (after the $\overleftarrow{v}$).

Page 11. This is more of a (minor) comment than a correction, and doesn't make much difference anyway. At the final step, the lop value of the final variable is not updated, as the algorithm stops. If instead it was updated, then the lop values record the order of last occurrence. Then in order to verify that the last occurrence words are equal, it suffices to verify that the final lop values are the same, which would make the earlier algorithm redundant.