Dear Editors,

We would like to thank the reviewers for the thorough reviews that they provided us with. Their comments proved valuable for improving our paper. In the revised version that we are submitting, we have changed the title and the structure of the paper to better separate the grammar description from the exploratory study of the datasets, which had been strengthened to explore specific research questions. Also, this version has an updated grammar description and diagrams to match the implementation, more detailed description of the dataset, updated grammar requirements and more detailed explanation for the selected parser generator.

Below we respond to the reviewers’ comments and we explain how we revised the paper to address them.

Looking forward to your feedback on the revised version of the paper.

Efthimia Aivaloglou

Associate Editor's Comments to Author:  
  
Editorial Board Member  
Comments to the Author:  
We have now received the reports of the reviewers (one from the previous round) and a new one. In light of their comments we recommend that you revise your manuscript to address the points mentioned in the reviews. In particular, we would like you to include information regarding Apache POI, as suggested by reviewer 2, explaining how your proposed grammar (and XLParser) performs in comparison to this other grammar.   
  
We hope that you will take the time to address all the issues raised by the reviewers and we look forward to your revised manuscript.  
  
Referee(s)' Comments to Author:  
  
  
**Reviewing: 2**  
  
  
*a) The “Conclusion” section is really tiny, and it does not present and ignore the main results described in section 5. Thus, I suggest to include the main results also in the conclusion (results highlighted in the boxes), exploring more the results that the abstract (the abstract is okay).*    
We agree, the Conclusion section needed an update. We added a description of the main results of the datasets analysis.

*b) In order to update the related work section, I strongly suggest to cite the paper  
  
Wensheng Dou, Shing-Chi Cheung, Chushu Gao, Chang Xu, Liang Xu, and Jun Wei. 2016. Detecting table clones and smells in spreadsheets. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016). ACM, New York, NY, USA, 787-798.*

We updated the introduction section and the related work sections with references to this paper, along with another relevant paper presented within 2016 by Cheung et al.  
  
  
*c) The paper claims that the previous research work ([7, 9, 10, 11, 12, 13]) would support the motivation to propose a new grammar for researching. However, it is a biased argument (confirmation bias), because the references 7, 9, 10, 11, and 13 are from the paper’s co-author, decreasing the motivation reliability. In addition, the paper use a single external reference ([12]) as a motivation, and despite the Badame and Dig ([12] paper’s authors) discussed, they have issues and found limitations when adopted Apache POI Java Library in their research, but they \*did\* not discuss exactly what kind of issues or limitations they had had using the tool. Consequently, all referenced work construct a weak foundation for motivating the paper.*

We strengthened the motivation for the grammar by including 3 recent works by other researchers that could have utilized a tested and published formula grammar instead of an undefined grammar or string comparison operations.

*Analysing the paper, I would highlight one major concern: ignoring/hiding previous grammar proposals and parser implementations.  
The paper abstract starts claiming “there does not exist a \*reliable\* grammar that is \*concise\* enough to facilitate formula parsing and analysis and to support research on spreadsheet codebases.” However, there is at least one grammar that the paper ignore completely: Apache POI ( [https://poi.apache.org/spreadsheet/formula.html](https://webmail.tudelft.nl/owa/redir.aspx?C=Whba2o_0RjgTkQ9jNYwgFS04yGoZhGxe80iQvUfyp3HbxPRVMmXUCA..&URL=https%3a%2f%2fpoi.apache.org%2fspreadsheet%2fformula.html" \t "_blank) ).  
In order to sustain the main claim of the paper, I strongly suggest to include an evaluation of Apache POI grammar and the POI tool, because XLParser is also a paper contribution. The Apache POI propose a grammar implementation on*[*https://apache.googlesource.com/poi/+/refs/heads/trunk/src/java/org/apache/poi/ss/formula/ptg/Ptg.java*](https://webmail.tudelft.nl/owa/redir.aspx?C=cv0UPYTt2Lwh-mkQV-qzPHzgTneqte_SlbF0XufwbLDbxPRVMmXUCA..&URL=https%3a%2f%2fapache.googlesource.com%2fpoi%2f%2b%2frefs%2fheads%2ftrunk%2fsrc%2fjava%2forg%2fapache%2fpoi%2fss%2fformula%2fptg%2fPtg.java)*.   
Apache POI is the current state of practice on open source Excel formula parsing (best of my knowledge). Consequently, the paper could be strongly motivated just showing that the proposed grammar (and XLParser) is better than POI, performing the same experimental protocol using POI, or at least making a table putting side-by-side POI and your proposal features. After that, you could also discuss XLParser advantages and weaknesses comparing on POI. Actually, reading the paper, it is just a “believing” that  XLParser could obtain a better success performance that Apache POI. What exactly are POI advantages and weakness comparing with the presented grammar?  
In the paper,  there is a paragraph  
“The above analyses are our main motivation towards defining a formula grammar. Having such a grammar will enable parsing spreadsheet formulas into processable parse trees which can in turn be used to analyze cell references, extract metrics, find code smells and explore the structure of spreadsheets. Essentially, a reliable and consistent grammar and its parser implementation, available to the spreadsheet research community, can support research on spreadsheet formula codebases and can enhance the understanding and usability of research results.”  
Thus, what is a piece of evidence supports the claim that the proposed grammar is more “reliable and consistent” or “concise” than Apache POI? Maybe XLParser is more reliable and consistent than Apache POI, but I could not find any evidence in the paper, because there is any evaluation of POI as a previous work.  
In the “spreadsheet formula grammar requirements”, I found a list:  
1. Compatibility with the official language  
2. Produce parse trees suited for further manipulation and analysis with minimal post-processing required  
3. Recognize the spreadsheet formula elements required for supporting spreadsheets research  
Consequently, before start a new proposal, probably the authors evaluated the POI features and identified that it does not follow the requirements all that requirements.  The concern is addressed just reporting that analysis. However, if POI is better (more “reliable and consistent” or “concise”) and the proposal grammar does not bring a new contribution, it means that the paper has a major issue.  
Thus, does Apache POI follow that requirements, or not? Why not? What kind of limitations are found in POI grammar that the proposal is able to address? What kind of features POI is better, equal or worst than XLParser?  
I suggest to clarify why POI grammar and parser is not able to support research needs as functions calls (of build-in and user-defined functions), function arguments, data (of different types) or references (to internal and external cells and ranges of different types). As illustrated on the Figure 1, maybe show a similar comparison using POI grammar, or performing the same experimental protocol using Apache POI, comparing parsing success and non-functional requirements as the time to processing the corpus or memory footprint, as examples.*

Indeed, we had omitted discussing the Apache POI formula parser. The reason was that its grammar is not published or defined anywhere, only the parser implementation is available. Still, we agree that it should be discussed. Based on your suggestions, on this version of the paper:

* We compared the parse trees produced by Apache POI to the ones from the proposed grammar
* We tested the Apache POI formula parser against the latest improvements that we had to include to our grammar in order to cover all cases found in the four datasets. We found 6 cases where Apache POI produces parse errors or incorrect parse trees, and we opened issues for those.
* We examined the drawbacks of using the Apache POI parser for research, and especially that the parser is marked ‘for internal use only’ and the parse trees are not exposed.

To discuss those issues, we added a separate section 6.1, comparing our grammar to the 2 alternatives (the official Excel formula grammar and the Apache POI parser). Moreover, we updated the introduction section to enrich the motivation towards our grammar based on this comparison.

**Reviewing: 3**  
  
Comments to the Author  
The paper presents a grammar for spreadsheet formulas and validated the parser with four large datasets. The problem is interesting and the paper is mostly easy to follow.   
  
Motivation for making the spreadsheet grammar is not clear. They authors noted, they needed a grammar (Section 3) for their previous and ongoing research (in the intro they do have some use cases). If that is required for a research purpose, this paper should have been a side work or part of that research. What is the use case for making such grammar? I find Section 3 is unprofessional, in particular the first sentence. I suggest the authors make it clearer about the use cases of making the grammar. As a software engineering researchers, how much do I care about this work?   
  
In building the grammar the authors followed the trial and error approach without showing any further meaningful ways to generalize the grammar. With such an approach, the current grammar might be incomplete for any new dataset. This was also evident from their experiment of new dataset. I am not convinced that such an approach is actually meaningful. The authors should have attempted to find a generalized approach for building the grammar. What did we learn from this paper in grammar construction? what is the novelty actually?  At least how good the proposed parser is? can you guarantee whether this is a complete and correct one? What is the take home?  
  
The paper does not have any threats to validity section. Are they sure that there have been no threats to the validity of the approach? I only see something about compatibility with the official Excel grammar. There seem to be other threats in the data collection, analysis and the nature of data as well. It would be great if the authors point out of such Internal Validity, Construct Validity, External Validity,  Reliability Validity and so on.   
  
The Related work section is incomplete. It only talks about studies with Spreadsheet/Excel datasets. I thought the paper is largely on the category of building grammars. Unfortunately, I don't see any single citations in building grammars in general. This seems to me a big weakness since as a reader I am not sure how people usually write such papers and valuate the proposed approaches. This is even more important for this paper, since they follow a trail and error approach. Is this a common approach for building grammar? Can they justify?

Comment 1.1 *Although the use case is notable (spreadsheet formulas) the described language itself is not. We are supposed to know how to create grammars for expression languages. If this is not the case for the current language, then you have to convince what theory or technology is lacking to construct the current parser. Is there a problem you are solving with this parser? What is the novelty? The paper does not convince me of a technical or theoretical issue of significant interest to warrant research.*

Answer 1.1.Our intention with this paper it to contribute to the spreadsheet research community, not the language engineering one. The problem that we are addressing with the parser is that there are numerous works in the spreadsheet research field that analyze spreadsheet formulas using either simple grammars which have not been evaluated and often contain obvious errors, or using implied, undefined grammars. We believe that a reliable grammar and its parser implementation, available to the spreadsheet research community, can support spreadsheet research and can enhance the understanding and usability of research results. Our motivation for publishing the parser is better explained in the second and third paragraphs of the introduction section of the paper.

Comment 1.2 *The claim that this grammar/parser is "bulletproof" is both vague and not substantiated. Parsing a corpus can be done by any grammar which over-approximates the structure and words of a language. If a parser can parse a corpus without crashing or producing parse errors, then you have done a first engineering step. Well done, but so what for the advance of software engineering? The next engineering questions are if the parser is precise (does it reject sentences which are not in the language?), if it is correct (does the structure reflect the semantics of the language) and is its definition unambiguous (does it uniquely define the structure of the sentences?). Neither of these questions are addressed by the paper with sufficient rigor, not conceptually, not in the experimental method, and thus not in the results. A contribution might be if (new) methods for answering these questions would have been explored and validated or compared to existing methods.*

Answer 1.2We agree that the bulletproof claim is too bold, and we removed it in this version. Also, in **the requirements that we set for the parser we added one against over-approximation, that it must recognize the spreadsheet formula elements that are required for supporting spreadsheet research.** The rejection of invalid formulas and the correctness of the parse trees are discussed in sections 6.3 and 6.4. We consider it, however, **out of the scope of this work to further explore, validate and compare methods for answering those questions.**

*Comment 2.4 In section 3, the authors present three requirements that they want the formula grammar to fulfil to be “bulletproof”. The requirements seem legitimate yet #3 is not. First of all, “be compact \*enough\* to \*feasibly\* implement” is a super vague statement based on two imprecise, unmeasurable parameters. Second, parser generators are used mostly for the reason of overcoming this obstacle and reliably generating code based on \*any\* number of grammar elements, so having hundreds of production rules is not a show stopper — on the contrary, it is a very good reason to use it. Thus, disregarding official formula grammars on the basis of their size is wrong and misleading for the readers: in reality you have probably disregarded them for reasons of quality. There is nothing fundamentally wrong about it, and grammar recovery, adaptation and debugging are far from trivial and/or industrialised, so a decision to develop a new grammar is sensible, especially if supported by a large testing endeavour, but it has to be honestly explained. The second reason given in the same introductory paragraph to section 3 is closer to the substance, but again, if the entire problem was solvable by inlining a bunch of superficial nonterminals, the authors would have done that with any grammar manipulation framework of their choice.*

Answer 2.4 We agree, requirement #3 was both too vague and misleading and we removed it from the paper. The official grammar is essentially disregarded because of its different purpose. **We wanted a grammar that is, for our purpose of doing spreadsheets research, “as simple as possible, but not simpler than that”, a point that we made in our presentation in SCAM but we had not explained sufficiently in the paper.** We updated section 3 to explain this. We also added a requirement that we believe must be made explicit, **that the grammar should recognize the spreadsheet formula elements that are required for supporting spreadsheets research (a requirement against oversimplification).**