Reviewer 3
Next, we provide answers (in blue text) to the comments/remarks pointed out by the reviewers (in black text).

Thanks for the careful reading and for the comments and suggestions. Below please find how we deal with them.

- This paper presents the syntax and semantics of a feature modeling language with probabilistic information on the features. Probabilities may be introduced for choices amongst different features and for optional features. The paper provides an operational and denotational semantics forThe proof of the following result is an immediate consequence of Lemmas~\ref{lem:pref}--\ref{lem:excl} (see Appendix~\ref{app:proofs} of the paper) their language and demonstrates their equivalence. The denotational semantic is implemented in Python and evaluated on feature models generated by BeTTy to show the applicability and scalability of the approach.

General comments:

The paper is well written and accessible. The mathematical notations are explained with the right level of details. A small example of a feature model with probabilities encoded using the defined language would be a nice plus.

My main concerns are about: (i) the empirical study and (ii) the absence of related work.

(i) This paper defines a new feature modeling language for probabilistic feature models but I did not get where are the probabilities in the feature models generated by BeTTy. Did I miss something? How does the probability help to be "several times more efficient than traditional analysis" if there is no probability in the feature model? Do you assume default probabilities for the choices and optional features?

In order to provide a clear description of this experiment, we have included – at the end of section 6.1 - a detailed description to clarify the main difference between the probabilities used to configure BeTTy (input parameter) and the obtained results that represent the probability of having a feature in a final product.

In the previous version of the paper we did not provide a clear description to explain why the probabilistic analysis makes easier the computing of the feature models. In our specific case, it is achieved by hiding sets of features, that is, those “hidden” features do not affect the processing of the probability of a feature X for being part of a valid product.

We compare the process of calculating if a given feature is part of a final product using the implementation of the probabilistic extension and the implementation of the denotational semantics from SPLA. In this case the experiments compare processing time using models containing different number of features. The obtained results show that using the former, we require less time to process large models.

The feature models used in the empirical study have been generated using BeTTy using different configurations. Why did you use those configurations? Please explain where do the values come from.

In the new version of the paper we explain why these configurations have been used in the performance analysis (see section 6.2). In essence, the idea is to show the impact of the Conjunction relationship on the time and memory required for processing the models.

Why is there only one configuration for the applicability, compared to the four used for the scalability? How confident are you that those feature models represent real feature models? For instance, do the parameters of the configuration mimic properties of real feature models? See for instance: Johansen M.F., Haugen Ø., Fleurey F. (2011) Properties of Realistic Feature Models Make Combinatorial Testing of Product Lines Feasible. In: Whittle J., Clark T., Kühne T. (eds) Model Driven Engineering Languages and Systems. MODELS 2011. Lecture Notes in Computer Science, vol 6981. Springer, Berlin, Heidelberg

In this work we focus on performance for processing models. We are interested on investigating the scalability of our approach when the model grows (increasing the number of features). Thus, we focus our study on worst case scenarios for modeling software product lines, where the major part of the cases are more complex to model than standard product lines, like the examples described in "Properties of Realistic Feature Models Make Combinatorial Testing of Product Lines Feasible".

I suggest reorganizing the section in:

- a small introduction to describe what the evaluation is about and what are the research questions. Please indicate what is measured and what do you consider as acceptable to consider the approach as acceptable and scalable;

In this new version of the paper we have re-organized this section. Also, we present some research questions to be answered at the end of this section.

- a setup subsection/paragraph describing the setup of the evaluation (including the generation of the feature models);

In this section we present two different experiments, where each one containing its own setup configuration.

- a result subsection/paragraph with the graphs and their description, and the answer to the research questions;

The results of each experiment, which include charts and tables, have been placed in the corresponding subsection of each experiment.

- a discussion subsection/paragraph with the comparison with the previous implementation and the threats to validity. Some of the questions here above may be discussed in that subsection/paragraph.

This section now contains the answer to the research questions and a discussion to compare the provided implementation with our previous implementation, which does not deal with probability.

Please note that since there is randomness involved (for the generation of the model and for the measuring the time analysis), the evaluation should be run a certain number of times to gain confidence in the results. Since the evaluation is not the core of the paper (and the data are no not used to derive precise conclusions), I think this can be avoided here. But, it has to be clearly described in the setup and discussed in the threats to validity.

(For more information about statistical significance for evaluation with randomness, see for instance: Arcuri, A. and Briand, L. 2014. A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Software Testing, Verification and Reliability. 24, 3 (2014), 219–250. DOI:https://doi.org/10.1002/stvr.1486.)

In the new version of the paper we have repeated each experiment several times. We include a table containing each obtained result – for the first experiment – and a box-plot for each configuration of the second experiment. In the latter, we generate and process 30 different models for each configuration.
The replication package is incomplete. It does not contain the BeTTy feature models and not all the data presented in the paper are there. Completing the readme with the requirements to run the scripts and providing a Makefile or bash script to run the complete evaluation would also be nice.

We provided [1] the scripts used in the empirical study of this paper, the current scope in the scripts used is not to develop a complete toolset, instead is to demonstrate the approach capabilities.

The package contains a README.txt files with each script description and some examples for executing them. They are all Python scripts.

[1]: http://ccamacho.github.io/phd/resources/03_splap.tar

(ii) There is a large amount of existing work on statistical analysis for SPLs and feature modeling with attributed feature models. Some of them are described at the end of the introduction. I suggest to move them to a related work section and to complete it. For instance:

Mahsa Varshosaz and Ramtin Khosravi. 2013. Discrete time Markov chain families: modeling and verification of probabilistic software product lines. In Proceedings of the 17th International Software Product Line Conference co-located workshops (SPLC '13 Workshops). ACM, New York, NY, USA, 34-41. DOI: https://doi.org/10.1145/2499777.2500725

Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. 2018. ProFeat: feature-oriented engineering for family-based probabilistic model checking. Form. Asp. Comput. 30, 1 (January 2018), 45-75. DOI: https://doi.org/10.1007/s00165-017-0432-4

Martin Fagereng Johansen, Øystein Haugen, Franck Fleurey, Anne Grete Eldegard, and Torbjørn Syversen. 2012. Generating better partial covering arrays by modeling weights on sub-product lines. In Proceedings of the 15th international conference on Model Driven Engineering Languages and Systems (MODELS'12), Robert B. France, Jürgen Kazmeier, Ruth Breu, and Colin Atkinson (Eds.). Springer-Verlag, Berlin, Heidelberg, 269-284. DOI: https://doi.org/10.1007/978-3-642-33666-9_18

The new version of the paper includes the citations to these works.

Specific comments:

- Section 1, "This cost may represent different aspects of a feature, such us lines" -> such as

Fixed

- Use 'cite' LaTeX package to automatically order citation numbers in the text.

Fixed

- Section 5, "The model used in the experiments has been executed several times" Please indicate how many times. Is it also the case for the models used for the scalability evaluation? If yes, please indicate how many times.

The total number of executions for each experiment has been indicated in the new version of the paper.

- Figure 4, are the values presented averages? If not, why not representing all the points and use a smoothing function like https://ggplot2.tidyverse.org/reference/geom_smooth.html to enhance the rendering?

The goal of the first experiment is to check how the implementation behaves, in the results presented in Figure 7 it's only displayed 1 model execution, thus we can't provides averages as each dot represent the time to compute each feature, we added a table with the first 10 executions of the same model.

Figures 10-15 show the set of all different types of models computed, in this case we generated models having from 1000 to 10000 features each and we generated 30 models of each. In this case we modified the operators we suspected that affects the most the computing time of the models, this is the parallel operator, so we used 3 configurations (probability of 0.01, 0.2 and 0.5) showing that this operator increases exponentially the computing power require to process the feature models.

- Section 5, "Secondly, an experiment" I suggest replacing experiment by evaluation.

Fixed