Dear Prof. Michele Loreti

We have addressed the concerns of the reviewers in this revised version of the paper: We have also adapted our paper to the stylistic guidelines and linguistic rules of the journal. Next, we provide answers (in blue text) to the comments/remarks pointed out by the reviewers (in black text).

Best regards,

Carlos.

Reviewer 1

Thanks for the careful reading and for the comments and suggestions. Below we describe how we dealt with them.

The paper presents SPLAP. It is a probabilistic extension of SPLA, a process algebra for specifying software product lines (SPL) presented in previous works by the authors

Details about each section

Section 1 Introduces the paper and mentions related approaches. But no comparison with other approaches is actually given.

In this version of the paper we have included a new section (see Section 2. Related work), which contains a comparison with current work that can be found in the literature.

Section 2.1 Introduces the syntax and operational semantics of SPLAP. The authors also show that the operational semantics of SPLAP agrees with the one of SPLA, if probabilities are discarded from the derivations. I believe that the description of the syntax delegates too much to previous works of the authors presenting the syntax of SPLA. More intuition should be given about the role and meaning of each operator. For example:

- How should conjunction be interpreted?

P \/ Q is the conjunction, intuitively it combines the products of both subterms P and Q by accumulating the features. There is an example (Example 2) in the paper to illustrate this.

- The operator 'P1 choose-one_p P2' allows choosing probabilistically among the two SPL P1 and P2 with probability p and 1-p, respectively. Then, it is said that this operator is actually n-ary. How should be interpreted such n-ary version? Which are the probabilities involved?

We have added more comments after Definition 1 and Examples 1 and 2 and 3 to explain the syntax. We have removed the comments about the choice being n-ary. We could have defined n-ary (by assigning a probability distribution to the n options), but we think the notation would have been more complex to follow.

Section 2.2 Introduces a denotation semantics for SPLAP, while **Section 3** proves that the two semantics are equivalent.

Section 4 Extends the language with a new operator to 'hide' in the semantics part of the features we are not interested in. This allows to obtain more compact state spaces, and hence to handle larger models.

I am not sure about this section for two reasons:

- I am not sure about the relevance/importance of this section in the paper. The presented notion of hiding is not considered/mentioned in any other section.

The hide operator is important for the implementation. It maybe not possible to compute all the SPL, it may be feasible to compute the probability of a particular feature. 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 mention this in the new version of the Introduction and we have included a paragraph with an example in the Empirical Study Section to explain it.

- Why do you need a new section for this operator? Couldn't you present it in section 2?

The hide operator is not a usual one in SPLs. So, we believe that a dedicated section is adequate in order to present the hiding operator.

Section 5 Validates the approach by running its tool instantiation against large SPL models. First, it considers a model with 1500 features generated using the tool BeTTy. It shows how to study the probability of having each of the 1500 features (and the runtime of this analysis). Then, the authors perform a scalability analysis of their tool for models with up to 10000 features. Three classes of models are considered, with different configurations in terms of presence of optional/mandatory features and choose-one/conjunction operators. I have two concerns about these analyses:

First, in the second-last paragraph of page 12, the authors justify unexpected variations in the runtimes in terms of disk latencies, memory paging, etc. These problems can be solved by performing each experiment many times (e.g. 10), and providing the average runtime for each experiment. In page 13 the authors mention that they repeated experiments more times, but apparently they chose one runtime.

The experiment described in Section 6.1 Model Analysis, has been repeated 10 times. A table containing the obtained results has been included in this section (see Table 1). Also, we clarify that the obtained results might not be only caused by the latencies (disk, memories, operating systems, etc..) but the stochastic nature of the generated models also has an important influence on the tendency of these results.

Additionally, for each performance experiment described in section 6.2, each feature model has been generated 30 times. The results are presented in a box-plot chart, to show the execution time for processing these models, and a line chart, for representing the average consumption of memory.

Second, I cannot understand how the models have been generated using BeTTy. In page 12 the authors say:

- the probability of having a mandatory feature is 0.2,The 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)
- … having an optional feature is 0.3,
- … having a feature in a choose-one is 0.25,
- … having a feature in a choose-one is 0.25.

To the best of my knowledge, a mandatory feature is a feature that must be present in every product generated by the product line. Hence, I would have expected to find that 20% of the features have probability 1 of being installed. However, as shown in figures 5 and 6, this is not the case. I believe that the authors should give more intuition about the nature of the models they consider.

This is a good point. Since this concept was not clearly described in the first version of the paper, we have re-written this part to clarify the values for configuring BeTTy and the obtained results.

In this case, it is important to differentiate the probabilities defined in BeTTy, which are used to generate a feature model, and the probability - calculated from the model - to have a feature in a final product. Let suppose that we configure BeTTy to generate a feature model using a probability of 0.2 for having a mandatory feature, which means that 20% of the generated features are mandatory. However, this does not imply that these features are part of the 20% of the generated products, because the probability of having a feature in a final product depends of the position of this feature in the model. For example, if a given mandatory feature is placed in a choose-one relationship, it is possible that the other branch (the branch that does not contain the mandatory feature) is used to generate the final product, discarding the branch containing the mandatory feature.

Discussion

In addition to the observations done above, I believe that the authors should improve the accessibility of their work. In particular, sections 2, 3, and 4 are long and contain many definitions and results. The authors should provide more motivation and examples. I would suggest the authors to use a running example starting from section 1 or 2, and use it to better explain/exemplify parts of SPLAP.

In the new version of the paper a new example has been added (see Section 3) to clarify how the general semantics works.

Furthermore, the authors show that the implementation of the denotational semantics of SPLAP is much more time/memory efficient (many orders of magnitude) with respect to the one of SPLA. But no explanation for this is given. In the first 3 lines of the conclusions, the authors say that the motivation behind the proposed probabilistic extension is to alleviate the state space explosion problem. Why is that? Why should the probabilistic extension alleviate such problem?

In this work, we refer as alleviating the state space explosion problem in the sense that large models can be efficiently processed in a reasonable time frame. In a previous experiment, our implementation of the denotational semantics from SPLA required around 300 seconds to process a model containing 17 features. However, in this paper we show that, using our implementation of the probabilistic extension, we are able to process – in less than 300 seconds - models containing 13.000 features.

Finally, given that this is a journal paper, I am surprised by the fact that proofs are provided in two appendices. Will these appendices be published? Will they be made available online?

We put the proofs in the appendix just for the sake of readability and we expect appendices to be published as well.

For these points, and for the ones raised above, I suggest the authors revise their submission.

Minor comments:

- Please rephrase the sentence 'The results are promising .... (450 featurThe 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)es).' I can't understand what it means.

This has been fixed in the new version on the paper.

- Among the related works, you did not mention ProFeat, a probabilistic language for SPL based on PRISM (https://link.springer.com/article/10.1007/s00165-017-0432-4)

This article is fairly recent and it has been cited in the new version of the paper.

- You mention three tools: QFLan, Z3, and Multivesta. I would suggest to provide a citation for each.

The new version of the paper includes these citations.

- Why do you forbid probability values 0 and 1?

These values give degenerate terms that are covered by other operators. We do not consider probability 0 in the optional feature because it would be equivalent to nil. We do not consider probability 1 in this operator because it would be equivalent to the ordinary prefix operator. It is true that we could have merged both operators, but we have preferred to follow the tradition of having both kinds of features: mandatory and optional. In the case of the choice operator, these values would give degenerated terms in which one of the sub-terms is never considered.

Reviewer 2

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

Summary

The paper under review introduces a probabilistic process algebra SPLA^P that extends SPLA, a process algebra for software product line (SPL) models that follows the feature-oriented domain analysis approach (FODA) [28]. SPLA has been specified by the authors in a preceding paper [1]. The authors claim that the use of their probabilistic extension of eases testing, analysis, and tailoring software products, e.g., by prioritizing feature combinations that are most likely to be included in the SPL. The operational semantics of process algebra terms is provided by a kind of probabilistic transition system. Compatibility to the semantics of [1] is shown when abstracting away probabilistic annotations. A case study is carried out using an own implementation of the formal framework.

Evaluation

The paper addresses an important field of SPL engineering, namely including stochastic information about features, their composition, and testing into the development process of SPLs. Unfortunately, the paper lacks of a justification of the probabilistic semantics, misses comparison to related work, and does not show feasibility of the approach. Hence I recommend to reject the paper.

Comments

1) There is no explanation or justification why the chosen operational semantics is suitable for its purpose. Intuitively I would expect that the probability attached to some A-transition corresponds to the probability of the event of composing feature A to the current feature combination. However, when, e.g., considering the term t1=A;tick /\ (B;tick /\ C;tick), after composing A, there is only a probability of 1/4 each for composing B and C. Hence, the probability of having the only possible feature combination ABC is less than 1, which let me conclude that the semantics is not appropriate. Furthermore, equality of process algebra terms is only considered with respect to their products. I argue that, as the semantics is an operational one, it would be more appropriate to take (probabilistic) bisimulation into account. The example term t2=(A;tick /\ B;tick) /\ C;tick then shows that the operational semantics is not associative (opposed to the claim that the proof of [1] takes over): choosing A in t1 has probability of 1/2, whereas it has probability of 1/4 in t2. The reason for these pathologies is in my opinion the somehow arbitrary choice of dividing probabilities in conjunctions by 2 (rules [con1], [con4] etc.). Please correct me if I am wrong, but in either case, there should be more explanations including examples where probabilities are involved (there is no example at all).

The transition system obtained with term operational semantics is “observed”, similarly as for classical trace (and not bisimulation) equivalence, only for the sets of features. The proof of the following result is an immediate consequence of Lemmas 3-12 (see Appendix A of the paper) select (each set being the labels of the transitions of a trace, thus abstracting also from trace label ordering) and the probabilities associated to them. For this reason, the conjunction operator is indeed associative: technically associativity is an immediate consequence of Proposition 3 and Definition 6 (an item about conjunction in both of them). We have rewritten the proof of Proposition 3 (including the case of conjunction, Lemma 8) more in detail trying to make it clearer. We also have included the example indicated by the reviewer in the paper. The 1/2 that is used in the operational semantics of /\ to assign probabilities to left-hand and right-hand branches is not critical: as a matter of fact we could have chosen any other distribution factor p in one branch and 1-p in the other branch (anyway both the

features on the left and on the right of /\ will have to be selected in the end). The choice of 1/2 w.r.t. any other p (with 0<p<1) just has the advantage that it makes, at least, commutativity of /\ obvious. In the example that the reviewer comments, the only product (set of features) of t2=(A;tick /\ B;tick) /\ C;tick is [A, B, C] with probability 1, that coincides with the set of probabilistic products obtained for t1=A;tick /\ (B;tick /\ C;tick). Additionally, we have put some examples of the operational semantics of terms in the paper.

2) Furthermore, for me it is not clear what the probabilities attached to transitions actually mean. The only modeling formalism for probabilistic SPLs is provided by SPLA^P terms. In [1] however, a translation of FODA diagrams to SPLA terms has been given. For me it is not clear how a corresponding translation would provide an SPLA^P term as the authors do not deal with probabilistic variants of feature models at all.

FODA does not have probabilities so their models are translated into SPLA [1]. In order to be translated to SPLA^P, it should be necessary to introduce the probabilities in FODA designs: in the optional feature (probability here determines the presence or not of such a feature, and corresponds to the frequency with which such a feature is selected) and in the choice operator (probability here determines whether the left-hand or right-hand features are selected, and corresponds to the frequency with which the corresponding selection is made).

3) There has been plenty of work done in the area of probabilistic SPLs and I miss a comparison with the existing approaches to estimate the contribution of the paper. For instance, the aim of [19] and the paper under review is highly related - what is the benefit of using the new approach presented in the paper instead of [19]? Furthermore, a reference to [a] is missing. There, a probabilistic feature model has been presented that supports dynamic (probabilistic) SPLs. I think that the intention of this paper is somehow covered by their approach as the stepwise feature compositions could be encoded into the (operational/automata-based) “feature controller” formalism in [a]. There is also an implementation of the theoretical framework of [a] published as [b] that could be mentioned.

In the new version of the paper, we have added a new section (Section 2) with an extended the related work as requested. Specifically related to the reference [29] one of the interesting aspects presented in our probabilistic extension is that any of the referenced research articles manage to describe in their work the use of multisets. Also, they do not explicitly work on the translation of FODA to represent probabilities and they do not introduce the notion of hiding those not needed features to calculate the probability of a specific feature.

4) The case study is not convincingly showing applicability of the approach. First, it is not clear how the randomly generated feature models include stochastic information. Correct me if I am wrong, but I understood that “the probability of having a mandatory feature is 20%” means that during the generation process of BeTTy the probability of generating a mandatory feature is 1/5. I could not find any hint where the annotated probabilities in the process terms come from and hence, I cannot evaluate the results. Furthermore, it is questionable to argue about runtimes of around 20ms - to make more reliable statements, the number of features should be increased to lower potential impacts of other processes or side effects.

This is a good point. Since this concept was not clearly described in the first version of the paper, we have re-written this part to clarify the values for configuring BeTTy and the obtained results.

In this case, it is important to differentiate the probabilities defined in BeTTy, which are used to generate a feature model, and the probability - calculated from the model - to have a feature in a final product. Let suppose that we configure BeTTy to generate a feature model using a probability of 0.2 for having a mandatory feature, which means that 20% of the generated features are mandatory. However, this does not imply that these features are part of the 20% of the generated products, because the probability of having a feature in a final product depends of the position of this feature in the model. For example, if a given mandatory feature is placed in a choose-one relationship, it is possible that the other branch (the branch that does not contain the mandatory feature) is used to generate the final product, discarding the branch containing the mandatory feature.
Additionally, we have repeated the experiment 10 times, where a new model is randomly generated and processed. Table 1 shows the obtained results from each execution.

Minor comments
- Definition 2.1 is missing (cf. Definition 5).

Fixed

- Reference [8] and [9] are the same.

Fixed

- Page 3 presents a long chain of references concerning probabilistic systems without any comment or explanation. It would be helpful to describe the relation of these publications to the presented paper.

Fixed

- Please explain in detail why the choose-one operator \/_p is n-ary. Furthermore, include the n-ary interpretation into your semantics (currently only binary).

We have removed the comments about the choice being n-ary. We could have defined n-ary (by assigning a probability distribution to the n options), but we think the notation would have been more complex to follow.

- The semantics is not uniquely defined. I assume that you look for the smallest probabilistic transition relation fulfilling the rules of Figure 2.

Fixed. We have introduced them properly in a definition.

- The statement that the rules of Figure 2 are in essential the same as in [1] is not correct as it is: the rule [req2] has been changed in [8] and taken from there.

The change in [8] was a minor change and we proved that both rules are equivalent: the change does not affect the semantics. We have preferred to keep such a change in this version. We have added a comment in the paper.

- Lemma 2: Q already fixed.

Fixed.

- Check typos in Proposition 5

Fixed.

References

[a] Clemens Dubslaff, Christel Baier, Sascha Klüppelholz: Probabilistic Model Checking for Feature-Oriented Systems. Trans. Aspect-Oriented Software Development 12: 180-220 (2015)

[b] Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, Christel Baier: ProFeat: feature-oriented engineering for family-based probabilistic model checking. Formal Asp. Comput. 30(1): 45-75 (2018).

These works have been cited in the new version of the paper.

-Reviewer 3

- 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