---------- REVIEW 1 ----------

This paper propose an approach for merging feature models based on the analysis of refinement semantics (decomposition/specialization/characterization) and variability types of features so that the merged feature model preserve the original refinement/variability relationships between features. The key idea of this merging operation is that every combination (union) of two product configurations (one from the first feature model and the other from the second) should be instantiated from the merged feature model, which is called as "cross-product semantics." The proposed approach is evaluated using several pairs of independently developed feature models.

* Points in favor
- The idea of cross-product semantics is very simple can be meaningful in some domains.
- Considering refinement semantics together with variability types in the merge can preserve original refinement semantics in the merged model.

* Suggestions for improvements

- Rationale for the approach is not explained, i.e. why the cross-product semantics is better than other approaches (union, intersection, etc.) needs to be explained. In section 6.1, Figure 9 shows how the proposed approach can be compared with others, but the example is not enough to convince that the cross-product semantics is better in general as which one is better largely depends on the problem on hand.
➔Section 6.1 has been updated to address the problem. In summary none of these kinds of semantics is perfect and the choice between them mainly depends on the problem:
(1) Choose union semantics when the original configurations have to be preserved.
(2) Choose intersection semantics when all the original constraints have to be preserved but the unique features are not needed.
(3) Choose cross-product semantics when the unique features along with original constraints between them have to be preserved but not the original configurations.
(Please see the paper for details.)

- The title of the paper is too general. Something like "Cross-Product Semantics for Merging Feature Models" might be more focused.
➔The title has been changed.

- In Fig. 6, because of the cross-product semantics, null configuration is missing. Whether this is acceptable or not depends on the domain. Also, the semantics of XOR is missing from the merge. Shouldn't that be converted to an Exclude constraint?
➔The original Figure 6 gives the example (the example in Figure 6 has been changed so I write it here):
(P with optional children A and B) + (P with xor-grouped children A and B) = P with or-grouped children A and B.
The reviewer asked if the result should be P with or-grouped children A and B, and A excludes B. That is, P with xor-grouped children A and B.

However, according to our cross-product semantics, the Exclude constraint cannot be added. Because:
[[P with optional A and B]] = { {P}, {P, A}, {P, B}, {P, A, B} }
[[P with xor-grouped A and B]] = { {P, A}, {P, B} }
[[P with or-grouped A and B]] = { {P, A}, {P, B}, {P, A, B} }.
So it can be clearly seen that the original example in Figure 6 is correct.
In addition, the missing of XOR also happens in the Union-semantic algorithms (where *optional + xor = optional*).

- Section 4.5 defines the rules for merging common features based on the variability types, but the rules listed in Table 2 is incorrect. For example, if feature B of the first feature model (Source 1) in Fig. 6 is mandatory, the merged feature model should be same as the first feature model as can be seen below:

[[Source 1]] = { {A, B}, {A, B, C} }
[[Source 2]] = { {A, B}, {A, C} }
[[Target]] = { {A, B}, {A, B, C} } = [[Source 1]]

The rule (Optional + Xor = Or) in Table 2 is only valid when both B and C are optional. When B is mandatory and C is optional, (Optional + Xor = Optional) as can be seen from the above example.
➔ We have adopted the reviewer's suggestion, and modified some of the rules in Table 2, Section 4.5. In order to preserve the cross-product semantics in all situations, two of the rules now satisfy the equation:

$[[Source\ 1]] \otimes [[Source\ 2]] \subseteq [[Target]]$,

instead of

$[[Source\ 1]] \otimes [[Source\ 2]] = [[Target]]$ (Strict cross-product).

In the first version of our paper, we have already defined the overall semantics of merging operation as $[[Source\ 1]] \otimes [[Source\ 2]] \subseteq [[Target]]$ (Section 3.3), therefore the modification will not affect other steps of the algorithm.
Besides, the examples in Figure 6 (Section 4.5) have also been modified to illustrate two situations satisfying "=" and "⊂", respectively.
(Similarly, the Union-semantic merging algorithms can be divided into Strict Union [2, 5, 7, 10] and Non-strict Union [1, 11] algorithms.)

- The rules of merging constraints in Table 3 also need thorough verification. For example, consider the rule No. 10 ("A requires B" + "B requires A") with two optional child features A and B of the common parent P, then cross-product will be:

[[Source 1]] = { {P}, {P, B}, {P, A, B} }
[[Source 2]] = { {P}, {P, A}, {P, A, B} }
[[Target]] = { {P}, {P, A}, {P, B}, {P, A, B} },

which means the target does not have any "require" constraints between features A and B. (So, the

rule No. 10 is not satisfied).
➔We have adopted the reviewer's suggestion and modified some rules in Table 3, with updated rationales explained in Section 4.6.

- In Fig. 9(c), the child features of Provider, Bits, Mhz, and Dual should be Xor-type, not mandatory.
➔The typos in Figure 9 have been corrected.

Fig. 4 shows distinct symbols (diamond, triangle, and double-lined cross) for different semantics of refinements, but these symbols are not used in the following models. Fig. 5 may be more understandable if these semantics symbols were used for explaining each merging rule.
➔The symbols have been added as suggested.

---------- REVIEW 2 ----------

This paper is well written althought I did not check its originality. It has a few typos. To start, its authors do not have numbers assigned to them to denote their belonging as well as numbers to the institutions they belong to. Please correct them.
➔They have been corrected. (For some reason, all the authors actually belong to the two institutions at the same time, so the authors have to be marked as "1, 2".)

---------- REVIEW 3 ----------

This is well-organized and -written paper. The proposed idea is well presented and discussed rigorously.

The weakness of the paper is the validation and evaluation of the method. The numbers of product features in section 5.1 are all less than 100, and so it is difficult to consider them as complex feature models. Also the cases of 'feature renaming' and 'parent compatibility' are not discussed explicitly. In practice, this might be the main bottleneck for merging feature models.
➔Section 4.8 has been updated to address the potential overhead of feature naming.
The data about feature renaming and clone features (effect of parent incompatibility) has also been included in Section 5.

Also, the refinement at section 4.2 is not new, but it only adapted the feature relationships (i.e., composed-of, generalization/specialization, implemented-by) in the FODA report. If there is any difference, it should be explained here.
➔The refinement semantics proposed in our previous work [12, 13] was inspired by FODA, and the fact is added to Section 4.2. The three kinds of refinements in our work can be considered as counterparts of the basic elements in UML class diagram (composition, specialization, and classes' properties). For the *implemented-by* refinements, we proposed a set of *complex constraints* to model such relationships. A complex constraint is a combination of a binary constraint (*requires, excludes*) and a variability predicate (*optional, all, xor, or,* etc.). For example, the relationship "a feature is *implemented by one or more* features" in FODA can be converted to the complex

constraint “a feature *requires or-grouped* features”. How to merge FMs with these complex constraints is a part of our future work.