Dear Editor and Reviewers,

This is the revision report for FAC-2024-0012.

We thank the reviewers for their comments. We have carefully revised the paper following these comments. The changes are highlighted in red in the revision.

The major changes include:

1. We use the task unboundedness/fragment container unboundedness problem and two real-world code examples (see Section 3) to illustrate why it is necessary to capture more precise information about activities and fragments in the models of multitasking mechanisms.

2. In Section 7.1, we relegate the code snippets (source code auditing) to the appendix to improve the readability.

3. In the second paragraph of Section 7.2, we explain the reason why we mainly use ValApp to validate the semantics of AMASS, instead of experimenting on a large number of Android apps.

4. In Section 8, we compare the abnormal behaviors resulting from the repeated execution of witness cycles and those by running the random testing tool Monkey, demonstrating a causal relation between task unboundedness/fragment container unboundedness and abnormal behaviors.

~~-------------------------------------~~--------------------------------

Below are the answers to the reviewer’s questions and comments.

**Review 1**

***Q1****. Although it is mentioned that "the existing formal semantics is far from complete" and the treatment of activities as atomic objects abstracts away internal structures like fragments, there is a lack of intuitive understanding for the reader about the problems this abstraction causes in static analysis or the advantages the proposed semantics provide over it. I recommend that the authors precede the formal definition with a real-world code example demonstrating these advantages. For instance, the example could illustrate what errors cannot be detected by static analysis methods based on existing definitions, why these errors are missed, and what structures or semantics should be considered to rectify this.*

**A**. We use the task unboundedness/fragment container unboundedness problem and two real-world code examples (see Section 3) to illustrate why it is necessary to capture more precise information about activities and fragments in the models of multitasking mechanisms.

***Q2****. Beginning from line 1969, or Section 6.2, the exposition by the authors becomes puzzling. They suddenly mention the selection of ten applications from F-Droid but do not discuss the criteria used for choosing these ten applications or justify why only ten were chosen. Furthermore, the rationale behind creating a new application instead of experimenting on a larger pool of existing applications is not made clear.*

**A**. In the second paragraph of Section 7.2, we explain the reason why we mainly use ValApp to validate the semantics of AMASS, instead of experimenting on a large number of Android apps. The main technical reason is that the semantic validation process is not fully automated, and thus experimenting on a large number of real-world apps becomes, unfortunately, too costly.

***Q3****. Whether a cross-verification method involving multiple individuals is used to ensure the reliability of manual confirmation of the task unboundedness/fragment container unboundedness issues.*

**A**. We do have 4 individuals whose results are cross validated. We have supplied the detail in the second paragraph of Section 9.2.2.

***Q4****. Demonstrate a causal link between the observed crashes and the task unbounded and fragment-container unbounded conditions.*

**A**. In Section 8, we compare the abnormal behaviors resulting from the repeated execution of witness cycles and those by running the random testing tool Monkey. We discover that among 74 unbounded apps that are reported by TaskDroid and confirmed manually, 14 (around 19%) apps end up with abnormal behaviors. This shows that compared to the random testing, the repeated executions of witness cycles indeed dramatically increase the chances of abnormal behaviors. The results give evidence of a causal link between the task unboundedness/fragment-container unboundedness and the abnormal behaviors of Android apps.

***Q5****. To substantiate the effectiveness of ICCBotAMASS over other methods, the authors should provide examples from real mobile applications, including an analysis of the code, to demonstrate that genuine task unbounded apps were indeed detected exclusively by ICCBotAMASS and not reported by other methodologies.*

**A**. In the last paragraph of Section 9.2, we elaborate on the explanation. ICCBotAMASS extracts models dynamically for those Android apps (especially the commercial ones) that cannot be decompiled by the other two methods, i.e., ICCBot and ActExtractor. For instance, the 9 commercial apps in Table 18 have been manually confirmed to be indeed task unbounded, but both ICCBot and ActExtractor fail to extract models out of them.

**Review 2**

***Q1****. Section 3.1 gives semantics of AMASS for Android 13.0, in comparison to that for Android 12.0 in FAC-2022-0049. However, no major changes are found. It is OK if the there is no change on Android's multitasking mechanism itself, but the paper should state this clearly.*

**A**. During the review process of this submission, Android OS is also under evolution. We follow the evolution in the revisions. Indeed, the semantics of AMASS for Android 13.0 is the same as Android 12.0. We state this fact in Section 4.2.1. Thank the reviewer for the suggestion.

***Q2****. Section 7 has been undertaken major changes with the details of Android OS auditing and app analysis. This part needs to be more concise though. In particular, the code snippets should move to appendix to keep a better readability.*

**A**. We relegate the code snippets to the appendix.

***Q3****. Evaluation on apps should include more that targets Android 13.*

**A**. We extend the static analysis experiments to include the apps for Android 13 (see Section 9.2.1 and Fig. 13).

***Q4****. In the revision, the authors should provide their response to the review of FAC-2022-0049.*

**A**. It has been attached at the end of this revision report.

***Q5****. The paper should conduct another literature review to study related work between the previous submission and this submission.*

**A**. We have conducted the literature review, but there is no more related work between the previous submission and this submission.

**Review 3**

***Q1****. One suggestion is regarding the reproduction of the application tool. In particular, the application tool, Taskdroid, is not made available, which may impede reproducibility.*

**A**. TaskDroid is open source and we added the GitHub site of TaskDroid in the beginning of Section 9.

***Q2****. A more detailed comparison with [HCW+19] would be even more helpful.*

**A**. At the end of Section 1, we elaborate on the comparison with [HCW+19].

**Revision report for the reviews of FAC-2022-0049.**

Dear Editor and Reviewers,

We thank the reviewers for their comments. We have carefully revised the paper following these comments.

The major changes include:

1. We follow the first reviewer’s suggestion by restricting the scope of this paper to the multitasking mechanism between activities and fragments.

2. We improve the presentation of Section 3 as follows.

(1) To simplify the definition of the semantics of AMASS𝐴𝐶𝑇,𝐼𝐹, we choose to skip the NOH flag in Section 3.1 (as a result, the definition of formal semantics is shortened from over 3 pages to around 1 page). The full semantics of AMASS𝐴𝐶𝑇,𝐼𝐹 is included in the appendix.

(2) Before the technical definition of the formal semantics, we use examples to help the readers get an intuitive understanding of the semantics first.

3. In Section 6, besides the semantics-validation experiments, we also audit the source code of the activity-fragment multitasking mechanism in Android OS to confirm the consistency of the formal semantics to the source code. The manual code audit and the automated semantics-validation experiments together increase the confidence in the correctness of the formal semantics.

4. We add discussions on the limitations of the dynamic model extraction in the end of Section 5. The effect of the dynamic model extraction on the performance of ICCBot\_{AMASS} is demonstrated by the experiment results in Tables 9 and 10.

~~-------------------------------------~~--------------------------------

Below are the answers to the reviewer’s questions and comments.

**Review 1**

***Q1****. I think the authors should better restrict the scope of this paper. Instead of claiming that they are formalizing the multitasking mechanism of Android, they should explicitly specify that they are formalizing multitasking between activities and fragments.*

**A**. We restrict the scope of this work to the Android multitasking mechanism between activites and fragments. Various parts of this paper, including the title, abstract, and introduction, have been adapted accordingly.

**Review 2**

***Q1****. I thank the authors for adding more details regarding the cross-version difference in the mentioned aspects. However, the current Section 3 is a bit unorganized. The authors should consider organize them into subsections or tables for better readability.*

**A**. To make Section 4.2 (semantics of AMASS\_ACT for the other versions of Android , Section 3.2 in the previous version becomes Section 4.2 in this version) more structured, we split Section 4.2 into subsections.

***Q2****. Please add the discussion on the limitation of using dynamic testing for model extraction, and clarify how this could affect the performance of ICCBot\_{AMASS}.*

**A**. We add discussions on the limitations of the dynamic model extraction in the end of Section 5. The effect of the dynamic model extraction on the performance of ICCBot\_{AMASS} is demonstrated by the experiment results in Tables 9 and 10.

**Review 3**

***Q1****. Improve the presentation of Section 3 to make it more understandable.*

**A**. We improve the presentation of Section 4 (Section 3 in the previous version becomes Section 4 in this version) in the following ways.

1) To simplify the definition of the semantics of AMASS𝐴𝐶𝑇,𝐼𝐹, we choose to skip the NOH flag in Section 3.1 (as a result, the definition of formal semantics is shortened from over 3 pages to around 1 page). The full semantics of AMASS𝐴𝐶𝑇,𝐼𝐹 is relegated to the appendix.

2) Before defining the formal semantics, we add examples to illustrate the semantics intuitively.

***Q2****. The correctness of the proposed formal model is difficult to judge. The validation of the formal semantics in Section 6 is insufficient.*

**A**. In Section 6, besides the semantics-validation experiments, we also audit the source code of the activity-fragment multitasking mechanism in Android OS to confirm the consistency of the formal semantics to the source code. The manual code audit and the automated semantics-validation experiments together increase the confidence in the correctness of the formal semantics.

***Q3****. The authors should discuss what the implications of their work in practice.*

**A**. We discuss the implications of this work in practice in the conclusion.

***Q4****. The authors should also discuss whether the proposed formal model is still relevant for later Android versions.*

**A**. We define the semantics of AMASS for the latest Android version (i.e. Android 13.0) in Section 4 and also discuss the differences in the semantics for the other versions (i.e. Android 6.0-12.0).