Dear Editor and Reviewers,

We thank the reviewers for their comments. We have carefully revised the paper following these comments and believe the quality of the paper has been greatly improved.

The major changes include:

1 We have rephrased the semantics of AMASS. In particular, in Section 3.1, we focus on Android 13.0 where we present the semantics in three steps with increasing complexity which we believe has improvised the readability. In later sections, we present the semantics for other versions of Android.

2 We have improved the presentation by giving more intuitive explanations and producing tables to summarize notations and abbreviations. We also included examples in various places to facilitate understanding.

3 We have carried out more extensive experiments for validating the formal semantics.

Below we respond to the major comments.

—---------------------------------------------------------------------------------------------------------

**Reviewer 1**

**Q1**: How is it different between the mechanisms to handle fragment transitions by AMASS and ICCBot?

**A**: Both AMASS and ICCBot collect information about fragment transactions. ICCBot does not record the calls to API addToBackstack(), so it misses the information mu in the transition rules A -mu-> T or F-mu-> T, where mu in {TS, NTS} denotes whether the fragment transaction should be pushed to the fragment transaction stack or not.

**Q2**: I cannot see the authors evaluating this novelty. It cannot be justified how the formulation of fragments can improve the state of the art. It would be better if the authors can demonstrate the advancements achieved by AMASS. For example, in the evaluation is it possible to add a baseline which does not analyze fragment transitions and another baseline based on ICCBot?

**A**: As suggested by the reviewer, besides ICCBot\_AMASS, we add two baselines ICCBot and ActExtractor (see Section 8.2.3), where ActExtractor is the model extractor from [HCW+19] and collects only the information about activities but not fragments. The experiment results are reported in Figure 11 where one can see that, when ActExtractor is used, TaskDroid cannot discover fragment container unbounded apps; on the other hand, if ICCBot and ICCBot\_AMASS are used, TaskDroid can discover 193 fragment container unbounded apps. Since both ICCBot\_AMASS and ICCBot collect information about fragments, they discover the same number of fragment container unbounded apps. Nevertheless, ICCBot\_AMASS discovers more task unbounded apps than ICCBot since the former can construct models using the dynamic approach, while the latter cannot.

**Q3**: I found it hard to evaluate whether the formal models are correct. The paper defines AMASS without explaining why the “multi-task mechanism” of Android should be modelled this way. What are the considerations behind these models? In particular, AMASS has different versions for different Android versions. How would the authors justify that the formalisms are correct?

**A**:  Fundamentally we model the multi-tasking mechanism from the external perspective, namely, to model its behavior (in logic, this is usually referred to as extensionality). It is standard to use transition system based formalism which has a history of over 70 years and we also adopt this methodology here. Technically, the multi-task mechanism requires to describe the evolution of different stacks. This key observation prompts us to put forward a model with multiple stacks which are widely used in the area.

In terms of correctness, we justify the sanity of the model based on empirical studies (cf. Section 6).

**Q4**: I have some doubts on the definition of “multi-tasking mechanism” in Android. It seems that the authors only focus on modelling inter component transitions based on activities and their enclosed fragments. Is this the definition of multi-tasking in Android? Before reading the paper, I thought the multi-tasking should refer to running two apps simultaneously by splitting the screen or start a service to run at the background. Would the authors better articulate the definition of multi-tasking?

**A**: We believe the reviewer’s intuitive understanding of multi-tasking is generally correct. However, when Android implements this mechanism, it is at the level of component, i.e., all of the high-level multi-tasking is implemented via (relatively) lower-level inter-component interaction/communication.

**Q5**: I also have concerns on the evaluation in Section 6. What was evaluated based on the app manually crafted by the authors? The section merely presents how the app was implemented and did not explain how it was used to validate the formal semantics. Even if the authors explained more details on this evaluation. I still have doubts on the setting of this experiment. The experiment can be overfitting. Basically, the authors are developing an app based on their own understanding on the multitasking or inter component transitions. This is definitely in line with their design on the AMASS models. At least, the authors may consider to ask some other people to build such an app for evaluation, or, may use available Android apps for validation.

**A**: We have completely rewritten the semantics validation part, included more details about the automated method of semantics validation, and carried out more experiments. In particular, we have used open-source apps (in addition to ValApp) to validate the semantics. One reason that we used the carefully designed ValApp to validate the semantics is that the open source or commercial apps are insufficient to cover the vast number of combinations of the features involved in the multitasking mechanism, e.g. launch modes, intent flags, activities and fragments, since they typically only include some widely used combinations.

**Q6**: Comparison with related state-of-the-art approaches: (Is there an adequate discussion of related work?): My major concerns on the comparison with related approaches is on the comparison with ICCBot.

**A**: ICCBot is only a tool to extract models from APK files, that is, to collect the (syntactical) information about activities and fragments. It does not define the semantics of the model. Moreover, we explain how ICCBot\_AMASS extends ICCBot for the model extraction in Section 4. Moreover, we also add the experiments to compare ICCBot\_AMASS and  ICCBot for the static analysis in Section 8.2.3.

**Q7**: The paper should be more self-contained. Some terms are not properly explained. For example, what is task affinity? This term starts to appear since the abstract but it is never explained. Also the authors do not provide any explanations for the four launch modes. What are the differences between them? Why are they important?

**A**: We have extended Section 2 to give a more specific intuitive introduction to all the features of the multitasking mechanism, including launch modes, task affinities, and intent flags. In Section 2, we add explanations for task affinities, which are of string data types, and are package names by default, but can be modified. For convenience, in the AMASS model (Definition 1), the task affinities are assumed to be natural numbers. We also add the intuitions of launch modes and intent flags in Section 3.1, before the description of the formal semantics.

—---------------------------------------------------------------------------------------------------------

**Reviewer 2**

**Q1**: Comparison with related state-of-the-art approaches: (Is there an adequate discussion of related work?): This is done in Section 9. I am not an expert in this particular field, but in general the discussions are rather brief, especially the paragraph "Static analysis" of Section 9. I encourage the authors to extend Section 9 to better reflect the SOTA and position their contributions more precisely.

**A**: We have extended the discussion of related work, especially on static analysis. We also have moved this part to Section 1.

**Q2**: Presentation

Q2.1 Rework on Section 3. In the current form, it is very difficult and challenging to understand the technical details and to appreciate why the formal model and its semantics are consistent with the actual operating systems.

**A**: We have completely rewritten Section 3. Now we presented the semantics step by step, following the idea of separating concerns. We first define the semantics for activities, then fragments. For activities, we define the semantics for launch modes first, then intent flags. We put the long definition of the semantics of the (complete) AMASS model in the appendix. We also add examples to illustrate the semantics. Our philosophy is that the main texts are easier but already sufficient for readers to understand the main ideas of the semantics, without diving into the (somehow) long and tedious semantics of AMASS models in the appendix. On the other hand, interested readers can still read the appendix for a complete understanding of the semantics.

*Q2.2. Sections 3.1.1-3.1.5 should be well explained, instead of just giving the definitions. I would suggest somehow integrate Section 3.2 into Section 3.1 to facilitate better understanding for the reader.*

**A**: We have rewritten Section 3 completely. In this revised version, we present the semantics step by step and add examples to illustrate the semantics. We also merged Section 3.1 and 3.2 as suggested.

**Q3**. Section 4: explain why the static and dynamic approaches for model extraction are correct.

**A**: It is hard to justify the correctness/completeness of the static approach in general, since it is rather a complex process and uses as a subprocess the decompilation of APK files, which by itself may already be imprecise. For the dynamic approach, we do know that it is incomplete since it constructs the models by running the app and executing the click events, which is normally just under approximation and may miss some transition rules. The main goal of this paper is to define the formal semantics of Android multitasking mechanism, validate the semantics, and use the semantics for the static analysis. We would like to say that the precision/correctness of the model extraction is somehow not central to the goal of this paper.

**Q4**. Section 6 is important to show the formal semantics can capture and are consistent with the actual behaviours. But the validations results are completely ignored (available at a given reference).

**A**: We have rewritten Section 6 and include more details of validating the semantics for activities and fragments. In particular, we use both ValApp and open-source apps for validating the semantics. Nevertheless, the results for validating the complete AMASS model are still too big to be included in the paper, thus they are included on a GitHub site.

**Q5**. The paper uses a quite lot of abbreviations, which are used for the formalisation in Section 3. This makes Section 3 very difficult to understand, as a reader has to go back to the beginning of Section 3 to check what those abbreviations stand for.

**A**: We add a table (Table 1) for the abbreviations and their full name, to ease reading.

—---------------------------------------------------------------------------------------------------------

**Reviewer: 3**

**Q1**: It would be good to elaborate the differences of the multitasking mechanisms among different version of Android, and highlight how they affect the behavior of launch modes, intent flags, task affinities, and structured activities with fragments, given that supporting multiple Android versions is claimed as a major new contribution compared with the conference paper.

**A**: We have a more detailed description of the differences of the semantics  for different Android versions. Moreover, we add an example to illustrate the main differences of the semantics.

**Q2**: The selecting 7 intent types of 21 is based on their prevalence. This seems not very convincing. It is good to consider having citations or pilot study to support this claim.

**A**: In the current version, we consider 10 out of 21 intent flags. We add explanations why the other intent flags are ignored.

**Q3**: ICCBot\_{AMASS} relies on dynamic testing, which may result in limitations with regard to coverage. While dynamic testing can provide precise information to construct the AMASS model, it is inherently limited that it only tests a subset of possible inputs and scenarios. As a result, the coverage achieved through dynamic testing may not be comprehensive or representative of all potential GUIs. This limitation needs to be discussed and evaluation.

**A**: We agree with the reviewer that dynamic testing only finds a subset of transitions. Nevertheless, we use dynamic testing for extracting models for those apps whose models cannot be extracted by static approaches. The dynamic testing is still valuable, even if it is incomplete. For instance, in the statics analysis, as a result of the dynamic testing, TaskDroid discovers more than 100 task unbounded apps than the purely static approach for extracting models (see Figure 11).

**Q4**: The dynamic testing procedure could be written into an algorithm for better understanding.

**A**: We have presented the pseudo-code of the model extraction algorithm by dynamic testing (see Algorithm 1).

**Q5**: Cross-app ICC is a key feature in Android's IPC. It would be good if this could be considered in the evaluation?

**A**: The AMASS model is general enough to take cross-app ICC into consideration since it does not care whether an activity is from the same app or not (what matters is task affinities, an app can use several affinities, while different apps can use the same affinities).

We take cross-app ICC into consideration in the model extraction process (see Section 4), where the information of multiple APK files involved in cross-app ICCs are provided manually. The statistics of models that are extracted from apps involving cross-app ICCs is put in Section 8.2.

**Q6**: Since multiple versions are supported. The paper is recommended to test those apps in different versions of Android, and check whether the semantics difference in each version may cause some issues in the tested apps.

**A**: We do experiments to test the impact of the differences of the semantics among different Android versions on the static analysis results (see Figure 9). The experiments show that the semantic difference does slightly affect the number of reported task unbounded apps. For instance, more task unbounded apps are reported on Android 6.0 than on the other versions.