Dear Editor and Reviewers,

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

The major changes include:

1. We rename the titles of Section 3.1 and Section 3.2 to explain the purposes of the examples.

2. We restructure Section 7.2 to explain why validating the semantics on a large pool of Android apps is hard.

3. In Section 9.2.2, we try to demonstrate the causality between the crashes and task/fragment-container unboundedness by analyzing the root causes of the reported crashes.

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

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

**Review 2**

***Q1****. The validation of formal semantics heavily relies on auditing Android OS source code. However, Android OS has been known complex. The paper should justify how quality is ensured in the auditing process to make it more convincing.*

**A**. We add the information how we ensure the quality of the auditing process: We apply a two-phase auditing process. The three authors audit the relevant parts of the source code of Android OS. Then the three authors have a joint discussion and achieve a consensus on the understanding of the source code.

***Q2****. The validation using self-designed app, i.e., ValApp, may cause questions on the reliability. The paper argues that "this way is largely credible since, after the options (e.g. launch modes and intent flags) are chosen, the rest of the validation process is the same as real-world Android apps". Does that mean ValApp only provides combination and sequence of interactive events? If so, can AMASS be extended to include an event scheduling to enable validation with real-world apps?*

**A**. We restructure Section 7.2 to explain why validating the semantics on a large pool of Android apps is hard and why the use of ValApp is necessary. ValApp provides combinations of intent flags or fragment transactions. Because ValApp is designed by us, we know how to generate the sequences of click events for transition rules of the AMASS model. The major obstacle for the semantics validation on real-world apps is the fact that it is hard to generate automatically the sequences of click events for transition rules in real-world apps.

**Review 3**

***Q1****. The addition of Chapter 3 significantly improves the clarity of the paper's motivation. However, the titles of Sections 3.1 and 3.2 appear to be somewhat arbitrary. I suggest renaming these sections to reflect the specific purpose or case studies they address, which would provide readers with a clearer understanding of their relevance to the overall paper.*

**A**. We rename the titles of Sections 3.1 and 3.2 to state the purpose of the two motivating examples.

***Q2****. Section 7.2 remains somewhat confusing regarding the selection of the 10 apps. The manuscript mentions the selection of 10 apps (line 1797) and later refers to 20 real-world apps (line 1890), but it is unclear how these groups are related. Are the 10 apps a subset of the 20, or are they entirely different sets? Additionally, while the authors explain that only 10 apps were selected because generating sequences of click events for transitions is still a manual process, this explanation is not fully convincing. The readers might struggle to understand why this process must be done manually and how labor-intensive it is, making it difficult to judge whether selecting only 10 apps is justified. I recommend that the authors reorganize this section to better clarify these points and provide more context on the manual process involved.*

**A**. We restructure Section 7.2 to explain why validating the semantics on a large pool of Android apps is hard.

***Q3****. I still do not understand why the authors do not demonstrate the causality between the crashes and task/fragment-container unboundedness by analyzing the root causes of the reported crashes.*

**A**. We use the ‘’adb logcat’’ command to extract the system logs when the abnormal behaviours appear. Moreover, we use the “adb shell dumpsys meminfo’’ command to extract the information of memory usage when the witnessing cycles are executed. In Section 9.2.2, based on these two types of information, we try to analyze the root causes of the reported crashes and demonstrate the causality.