SIG-Droid: Automated System Input Generation for Android Applications

تفاوت آزمون برنامه‌های کاربردی اندروید با سایر برنامه‌ها:

۱. برنامه‌های کاربردی اندروید وابسته به مجموعه ای از کتابخانه‌هایی هستند که در بیرون از ابزار یا شبیه‌ساز در دسترس نیستند. کد اندروید در Dalvik Virtual Machine(DVM) اجرا می‌شود. برخلاف کدهای برنامه‌های جاوا که در Java Virtual Machine(JVM) اجرا می‌شوند. پس به جای java byte-code برنامه‌های اندروید به Dalvik byte-code کامپایل می‌شوند.
۲. برنامه‌های اندروید بسیار وابسته به کتابخانه‌های چارچوبه کاری هستند و این موضوع باعث ایجاد مشکل واگرایی مسیر[۱]می‌شود. در اجرای نمادین اگر یک مقدار نمادین از زمینه[۲]برنامه خارج شود، مثلا برای انجام یک پردازش به یک کتابخانه داده شود یا در اختیار چارچوبه کاری قرار گیرد، گفته می‌شود که واگرایی مسیر اتفاق افتاده است. واگرایی مسیر موجب ایجاد دو مشکل می‌شود: الف) موتور اجرای نمادین ممکن است نتواند کتابخانه خارجی را اجرا کند پس تلاش بیشتری لازم است تا بتوان آن کتابخانه را نیز به صورت نمادین اجرا کرد. ب) در کتابخانه خارجی ممکن است تعدادی قید وجود داشته باشد که در خروجی و حاصل پردازش کتابخانه موثر باشند. از این جهت این قیدها در تولید موردآزمون‌ها موثر خواهند بود و به جای آزمون برنامه اصلی، تمرکز به آزمون مسیر واگرا شده در کتابخانه معطوف می‌شود و به طور پیوسته لازم است تا قسمتی از سیستم عامل اندروید به صورت نمادین اجرا شود که در کل موجب ایجاد سربار زیاد در آزمون برنامه می‌شود. یک مثال پرکاربرد از این نوع می‌تواند Intentها باشد که سیستم پیام‌رسانی بین مولفه‌های مختلف در اندروید است. به وسیله Intent یک مقدار به یک مولفه در درون یک برنامه یا به مولفه‌ای در برنامه دیگر ارسال می‌شود. Intent بعد از خارج شدن از محدوده برنامه وارد کتابخانه‌های سیستمی‌شده و بعد از آن وارد مولفه مقصد می‌شود.
۳. برنامه‌های اندروید رخدادمحور هستند. به این معنی که در اجرای نمادین موتور اجرا باید منتظر کاربر بماند تا با تعامل با برنامه یک رخداد مثل لمس صفحه نمایش ایجاد شود. علاوه بر کاربر برنامه‌های ثانویه هم می‌توانند رخداد تولید کنند مثل رخداد تماس ورودی یا دریافت یک پیام.

همان طور که در شکل هم دیده می‌شود، SIG-Droid از سه قسمت اصلی تشکیل شده است.

۱. Behavior Model: در قسمت از ابزار موردکاربردهای برنامه و رشته‌ای از رخدادهای مرتبط با هر یک استخراج می‌شود. برای این کار با استفاده از ابزار MoDisco(که ابزار استخراج گراف فراخوانی برنامه جاوا

---

[۱] Path divergence
[۲] Context

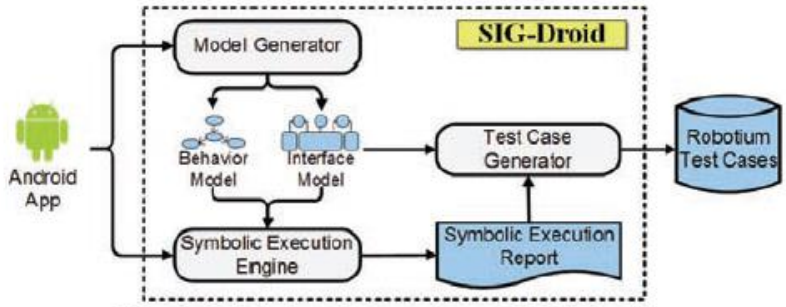
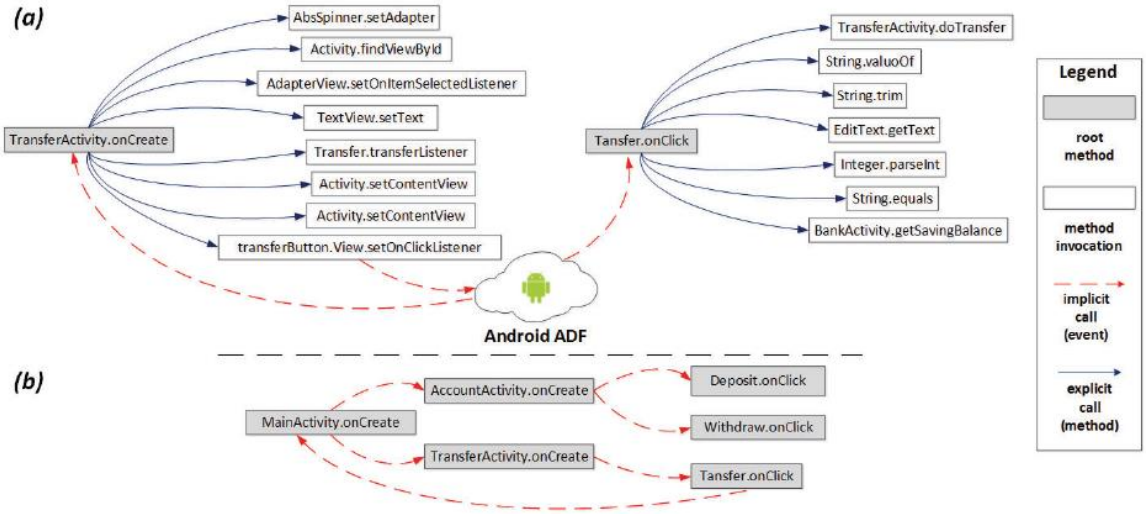


Fig. 3: High level overview of SIG-Droid.

هست) گراف فراخوانی توابع استخراج می‌شود. چون برنامه های اندرویدی با سیستم عامل بسیار در تعامل هستند گراف استخراج شده برخلاف برنامه های به زبان جاوا گراف های یک پارچه نیستند و از تعدادی زیرگراف تشکیل شده اند. SIG-Droid با بررسی کد گراف های استخراج شده را به هم وصل کرده و گره های میانی گراف را حذف می کند. نمونه ای از این گراف در زیر آمده است:



۲. Interface Model: در اندروید در کنار کلاس ها و کد هایی که به زبان جاوا نوشته می شوند، تعدادی فایل به زبان xml وجود دارند که مشخصات Activityها (فایل manifest.xml) و ساختار و چینش اجزای هر Activity در آنها ذخیره می‌شود. با تحلیل ایستای این فایل‌ها، ابزار اطلاعات مربوط به UI و اجزای هر صفحه (widgetها) استخراج می‌شود.

۳. اجرای نمادین: همان طور که در ابتدا گفته شد برنامه های اندروید سه تفاوت عمده با برنامه های عادی دارند که در اجرای نمادین باید به آنها توجه شود.

- برنامه های اندروید به جای java byte-code به dalvik byte-code تبدیل می شوند. پس برای اینکه بتوان از موتورهای اجرای نمادین مربوط به جاوا استفاده کرد، باید کدهای اندروید را با کامپایلر جاوا کامپایل کرد. اما همان طور که گفته شد، کتابخانه‌های اندروید در جاوا وجود ندارند

و عملیات کامپایل نیاز دارد که این کتابخانه ها و فراخوانی به آنها با تعدادی کلاس (stub) شبیه سازی شوند.

- مشکل دیگر واگرایی مسیر هست. برای حل این موضوع کتابخانه های ساختگی(mock) ایجاد می شوند که تنها یک مقدار دلخواه به عنوان خروجی تولید می کنند. با این کار ابزار درگیر آزمون کتابخانه های خارج از برنامه نمی شود.
- برای آزمون جنبه های مختلف یک برنامه لازم است تا رشته ای از رخدادها تولید شود. برای این کار با استفاده از Behavior Model تعدادی کلاس Driver برای این کار نوشته می شود. برای تولید رشته های مختلف در این ابزار گراف BM با روش DFS پیمایش می شود.
- موضوع آخر مشخص کردن ورودی های نمادین برنامه هست. به این منظور با استفاده از Interface Model و بررسی ویجت‌های مختلف ورودی های نمادین تعیین می شوند. مثلا اگر در برنامهد یک TextBox وجود داشته باشد، تمام متغیرهایی از کد که مقدار ورودی در این TextBox را در خود ذخیره می کنند به عنوان متغیر نمادین در نظر می گیرند.

۴. مولفه تولید موردآزمون: با استفاده از Interface Model و گزارش موتور اجرای نمادین تعدادی مورد آزمون برای اجرا به وسیله Robotium تولید می‌شود.

هدف اصلی SIG-Droid پوشش هر چه بیشتر مسیرهای مجود در برنامه است. برای اندازگیری این مورد هم از ابزار متن‌باز EMMA استفاده می کند. موتور اجرای نمادین JPF (Java Path Finder) هست که برای این کاربرد تغییر داده شده است.