Typestate Analysis for Android Applications

Ashish Mishra, Y. N. Srikant, Aditya Kanade

Indian Institute of Science

1. Android Control Flow

- Complex control flow structure.
- Asynchronous calls and call backs using IPC binder, called Intent.
- Asynchronous calls from System Services.
- Component life cycles and event handler call backs, to optimize resource usage and user experience.

2. Typestate

•**Quoting Storm and Yemini-**.

"type- determines the set of operations ever permitted on an object, the typestate determines a subset of these operations permitted in a particular context"

• Example, *Iterators* over a Collection object, *next* permitted only if *hasnext* is true.

3. Motivation

- Android resources and other APIs have complex usage protocols.
- Violations are difficult to trace and hard to debug.

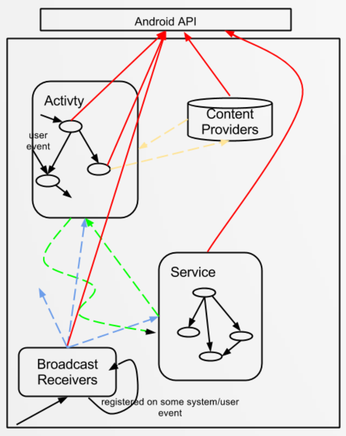


Fig 1. ICC in an Android app

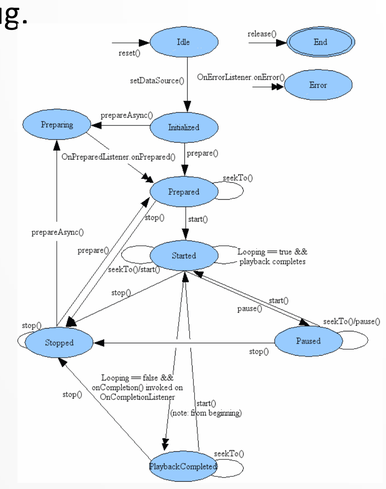


Fig 2. Protocol for Android MediaPlayer API

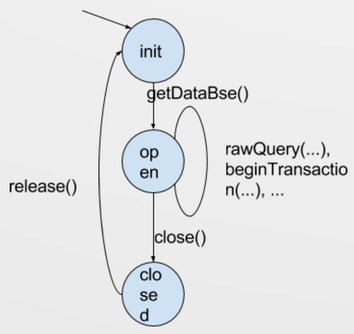
source- https://developer.android.com

•Hard to statically analyze, due to complexity at two levels control flow and typestate protocols.

4. A Database App

```
1  FirstActivity extends Activity {
2      SQLiteDatabase mydatabase = null;
3      onCreate(Bundle savedInstanceState) {
4        myDBhelper = new MyDBHelper(this);
5        mydatabase = myDBhelper.
             getWritableDatabase();
6   Intent intent =
7  new Intent(this, DataBaseActivity.class);
8        startActivity(intent);
9      }
10     onStart() {
11       Cursor resultSet =
12  mydatabase.rawQuery("Select_*_from_myTable",null
         );
13   [...]
14     }
15     onPause() {
16       mydatabase.close();
17     }
18   }
```

```
19 DataBaseActivity extends Activity {
20   onCreate(Bundle savedInstanceState) {
21   FirstActivity.mydatabase.beginTransaction
         ();
22   FirstActivity.mydatabase.close();
23   //ERROR
24   }
25 }
```



Listing 1. DataBase example app

• Precise and correct modelling of asynchronous calls and life cycle call-backs required.

5. Our Approach

- A precise and correct modelling of Android Asynchronous control flow, call backs and Component life cycles.
- Generating AICFG , an asynchronous Inter-procedural Control Flow Graph for the application.
- A flow insensitive whole app , alias analysis to soundly track state changes.
- A flow and context sensitive whole app, Typestate analysis, using graph reachability based inter-procedural analysis.

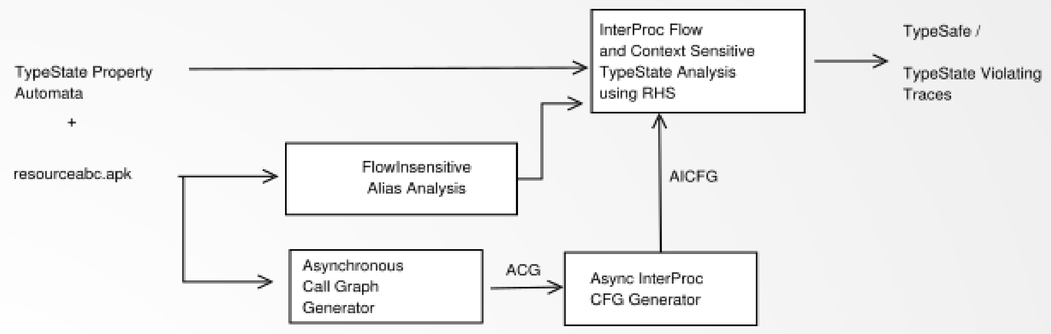


Fig 3. Flow Diagram for Typestate analysis

6. Results

- First Typestate analysis over Android apps.
- Added 10 new test benchmark apps to DroidBench android static analysis suite.
- Analyzed typestate properties of important resources like Camera, File, MediaPlayer etc.
- Analyzed Some real world Android applications and found typestate violations in them.
- Compared against a typeState analysis over Control flow model used by other works(IccTA , AmanDroid), giving lesser FPs , and increasing the TPs.

		Model Based Analysis				IccTA based Analysis			
AppName	Actual Violations	Violations found	TP	FN	FP	violations found	TP	FN	FP
Camra API	1	1	1	0	0	1	0	1	1
MediaPlayer API	2	2	2	0	0	1	0	2	1
SQLite API	3	4	3	0	1	1	0	3	1
DataBases	2	2	2	0	0	1	0	2	1
Files	1	1	1	0	0	1	0	1	1
SocketsFiles	1	1	1	0	0	1	0	1	1
Streams	2	2	2	0	0	1	1	1	0

Table 1. Typestate analysis results on test apps

7. Conclusion

- Gave a first precise and correct model for the Asynchronous control flow, life cycles and ICC in Android apps.
- Performed the first, sound Typestate analysis over android apps and compared the results against the control flow semantics used by other Android static analysis works.
- One limitation occurs due to the use of RHS for the analysis, which does not scale for big programs, increasing scalability is one future direction we aim at.

8. References

[1] L. Li, et. al. IccTA: Detecting Inter-Component Privacy Leaks in Android Apps. In Proceedings of the 37th International Conference on Software Engineering (ICSE 2015), 2015 .

[2] F. Wei et. al. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, pages 1329–1341.

[3]Thomas Reps, et. al. 1995. Precise interprocedural dataflow analysis via graph reachability. In Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '95).

Contact email: ashishmishra@csa.iisc.ernet.in