DMMPP

Constructing Dummy Main Methods for Android Apps with Path-sensitive Predicates

Baoquan Cui **Rong Qu** **Jian Zhang**

崔保全 瞿荣 张健

Publicly Available at https://github.com/cuixiaoyiyi/Leopard

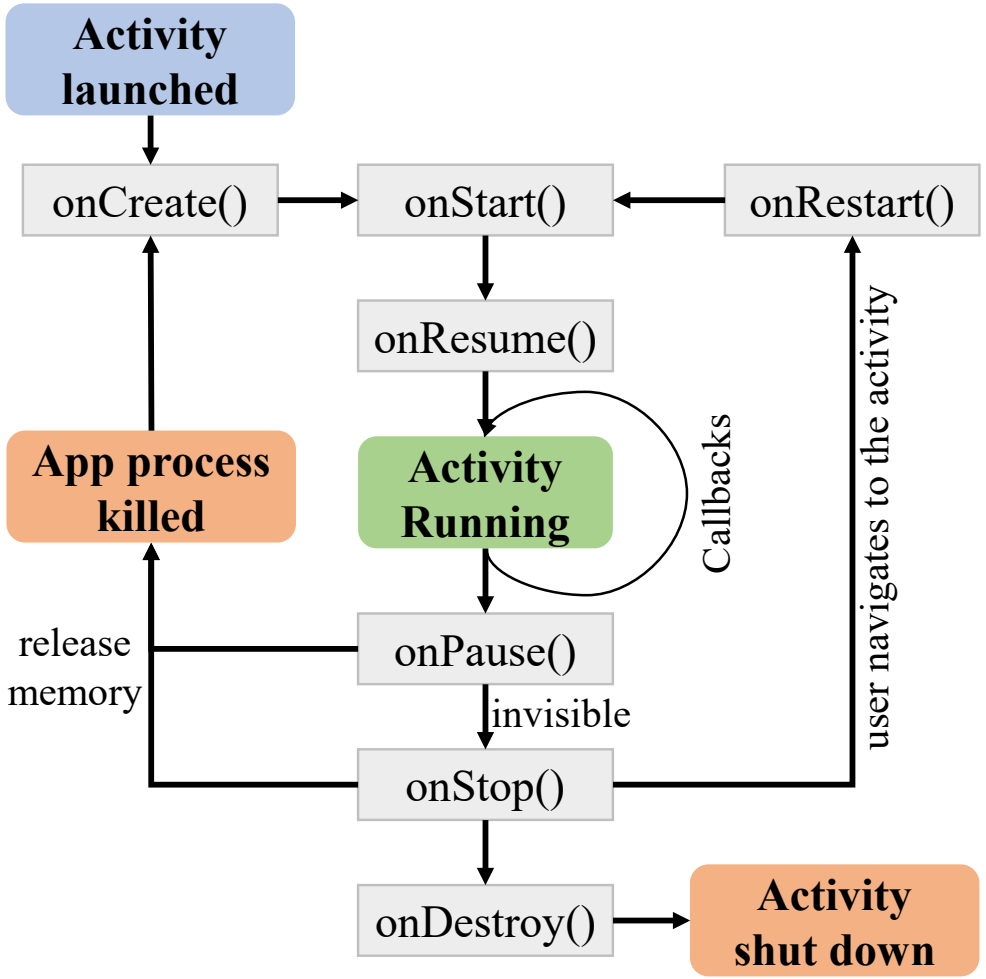
Institute of Software, Chinese Academy of Sciences (ISCAS)

中国科学院软件研究所

Background: Android Entry Method

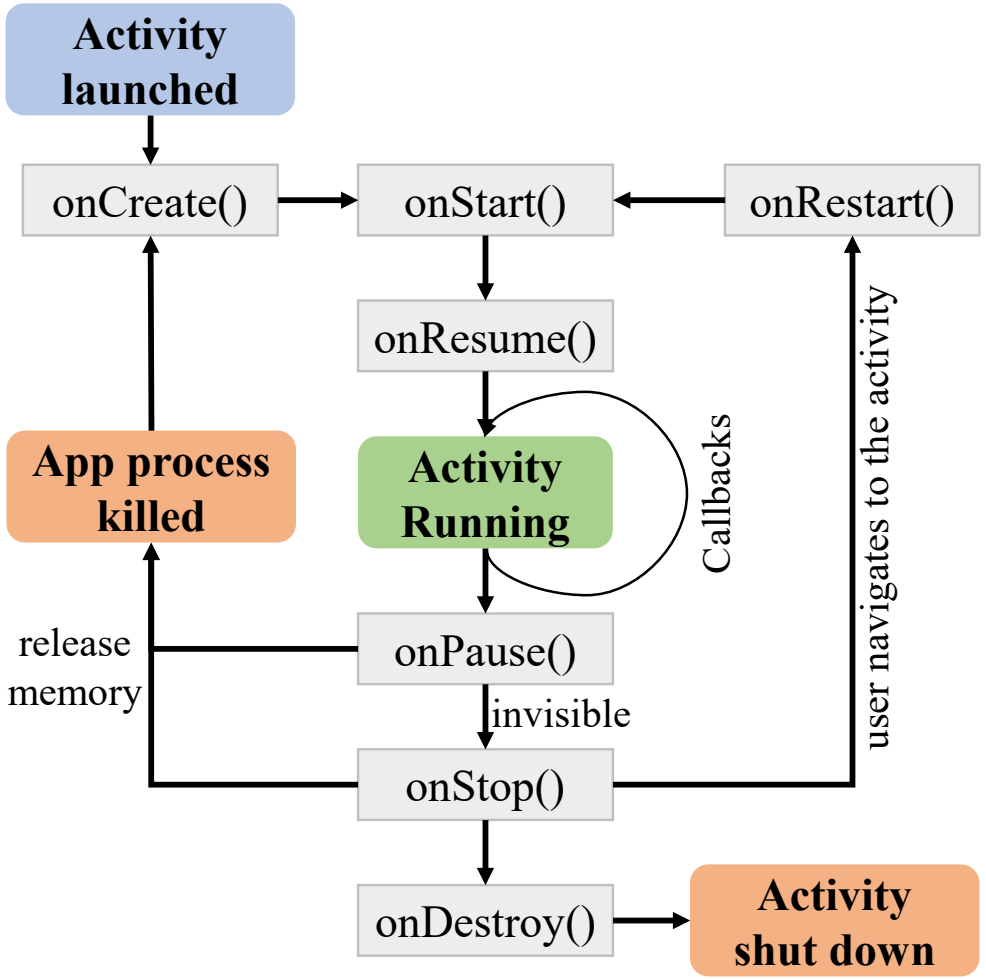
- Android application lacks of the main method

Background: Dummy Main Method

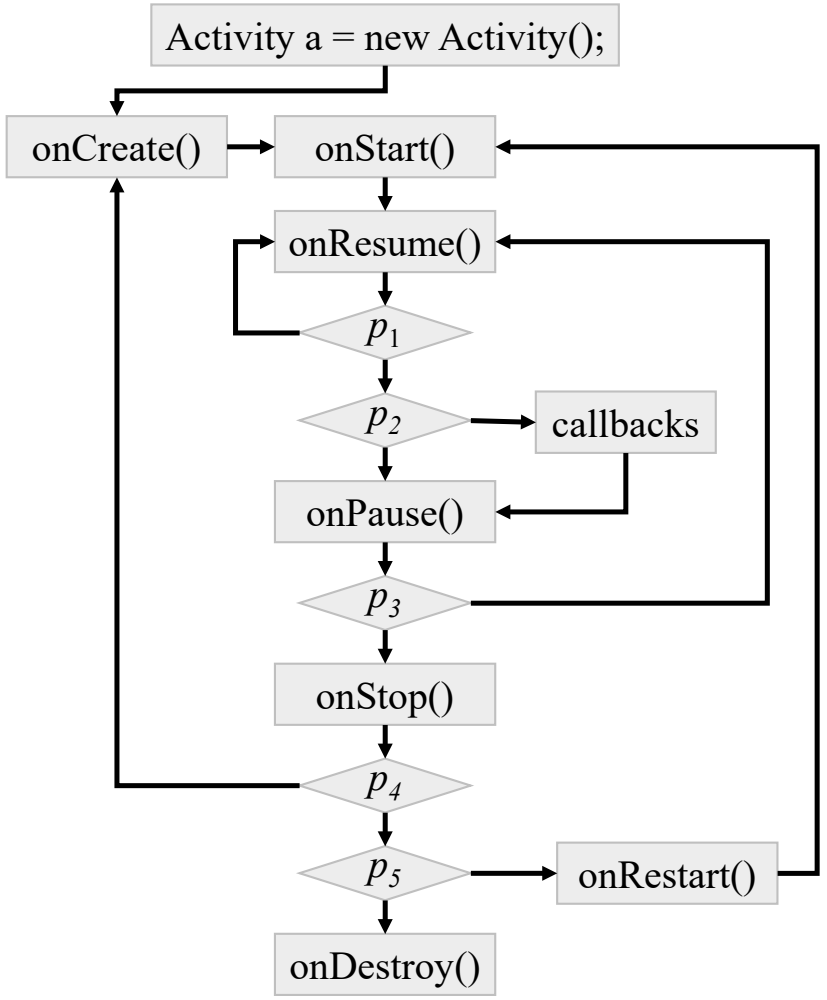


Lifecycle of Activity

Background: Dummy Main Method



Lifecycle of Activity



Control Flow Graph of Dummy Main Method of Activity

Background: Dummy Main Method

- The dummy main generated by FlowDroid does not consider whether the path condition is satisfied
  - path-insensitive
- However, for path-sensitive analysis, it can affect the analysis accuracy
  - false positives and false negatives.

Motivation Example

```
// Path-insensitive Predicates modelled by FlowDroid
void dummyMain(){
    int i = 0;
    if(i == 0){ // p1 : i == 0
        onCreate(...);
        if(i == 1){ // p2 : i == 1
            if(i == 2){ // p3 : i == 2
            onResume();
            ...
}}}}
```

Motivation Example

```
// Path-insensitive Predicates modelled by FlowDroid
void dummyMain(){
    int i = 0;
    if(i == 0){ // p1 : i == 0
        onCreate(...);
        if(i == 1){ // p2 : i == 1
            if(i == 2){ // p3 : i == 2
            onResume();
            ...
}}}}
```

- Path condition:

$P_1(i==0) \wedge P_2(i==1) \wedge P_3(i==2)$

Unsatisfiable

onResume(): unreachable

```
void dummyMain(){
    int i = 0;
    if(i == 0){ // p1 : i == 0
        onCreate(...);
        if(i == 1){ // p2 : i == 1
            if(i == 2){ // p3 : i == 2
            onResume();
            ...
}}}}
```

- Path condition:

$P_1(i==0) \wedge P_2(i==1) \wedge P_3(i==2)$

Unsatisfiable ✗

onResume(): unreachable

Motivation Example

// Path-sensitive Predicates Generated by DMMPP

```
void dummyMain(boolean[] bArr)
    if(bArr[0]){ // p1 : bArr[0]
        onCreate(...);
        if(bArr[1]){ // p2 : bArr[1]
            if(bArr[2]){ // p3 : bArr[2]
                onResume();
                ...
}}}}
```

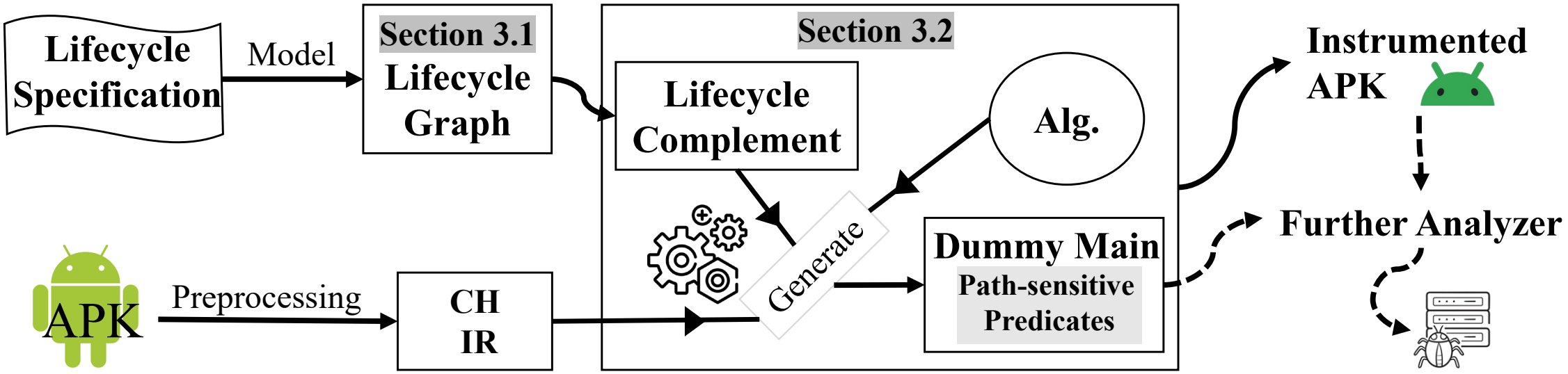
- Path condition:

$$P_1 \wedge P_2 \wedge P_3$$

Satisfiable

onResume(): reachable

Overview of DMMPP



Component Lifecycle Graph (CLG)

- A component lifecycle graph is a directed graph, denoted as

$$\mathbf{CLG} = \langle \mathbf{N}, \mathbf{E}, \top, \bot \rangle$$

  - N is a set of nodes and a node represents a lifecycle method, a callback list or a nop instruction
  - E is a set of edges
  - $\top$ is the start node
  - $\bot$ the end node

Lifecycle Method Complement

```
+ void onDestroy(){
+        super. onDestroy();
+}
```

Construction

Name: $\frac{A \in (a...zA...Z\$<>)*}{\textbf{name}(A)}$

Type: $\frac{\textbf{name}(T)}{\textbf{type}(T)}$

Variable: $\frac{\textbf{type}(T) \quad \textbf{name}(V)}{\textbf{var}(T, V)}$

ArrayType: $\frac{\textbf{type}(T)}{\textbf{arrayType}(T)}$

Expression: $\frac{\textbf{instruction}(E)}{\textbf{instruction}(\textbf{exp}(E))}$

Return: $\frac{-}{\textbf{instruction}(\textbf{returnVoid}())}$

Program: $\frac{\textbf{instruction}(I) \quad \textbf{program}(P)}{\textbf{program}(I,P)}$

New: $\frac{\textbf{type}(T)}{\textbf{instruction}(\textbf{new}(T))}$

IfStmt: $\frac{\textbf{exp}(E) \quad \textbf{type}(B)\ is\ boolean}{\textbf{instruction}(\textbf{ifStmt}(B,E))}$

Goto: $\frac{\textbf{exp}(E)}{\textbf{instruction}(\textbf{Goto}(E))}$

Load: $\frac{\textbf{var}(Index, ARR, Ret) \quad \textbf{ArrayType}(Arr) \quad \textbf{type}(Index)\ is\ integer}{\textbf{instruction}(\textbf{load}(Arr,Index,Ret))}$

Class: $\frac{\textbf{name}(Name) \quad \textbf{type}(Super) \quad \textbf{var}(Field) \quad \textbf{fun}(Sign)}{\textbf{class}(Name,Super,Field,Sign)}$

Function: $\frac{\textbf{name}(Sign) \quad \textbf{var}(Arg) \quad \textbf{program}(Body)}{\textbf{fun}(Sign,Arg,Body)}$

Invocation: $\frac{\textbf{var}(Base) \quad \textbf{name}(Sign) \quad \textbf{var}(Arg) \quad \textbf{var}(Ret)}{\textbf{instruction}(\textbf{call}(Base,Sign,Arg,Ret))}$

InsertExpr: $\frac{\textbf{exp}(E_1) \quad \textbf{fun}(Sign) \quad \textbf{exp}(E_2)}{\textbf{insertExpr}(Sign,E_1,[after],\textbf{site}(E_2))}$

Syntax

Algorithm 1: Dummy Main Method Generation

Input: *component*, *clg*

```
1  fun ← NULL, args ← ∅, body ← ∅, node ← clg.⊤, stack ← ∅;
2  caller ← NULL, map ← ∅; // map⟨Node, Expression⟩
3  stack.push(node);
4  caller ← instruction(new(type(component)));
5  // generate method invocations
6  while stack is not empty do
7      node ← stack.pop();
8      stmt ← NULL;
9      if node is ⊥ then
10         stmt ← instruction(returnVoid);
11     else if node is not ⊤ then
12         if node.m is nop instruction then
13             stmt = instruction("nop");
14         list ← getArgsFromParameters(node.m, args);
15         stmt ← instruction(call(caller,node.m,list,NULL));
16         map.put(node, call);
17     body.add(stmt);
18     map.put(node, stmt);
19     successors ← clg.successorOf(node);
20     stack.push(successors);
21 // insert if- and goto- statements
22 predicateIndex ← 0;
23 predicates ← getPredicatesFromParameters(args);
24 for node ∈ clg.N do
25     current ← map.get(node);
26     next ← body.getNext(current);
27     if node is branch node then
28         // crystal predicate
29         load(predicates, predicateIndex, predicate);
30         predicateIndex ++;
31         for succ ∈ clg.successorOf(node) do
32             if next is not succ then
33                 next ← succ;
34         ifStmt ← instruction(ifStmt(predicate, next));
35         insertExpr(body, ifStmt, "after", current);
36     succ ← clg.successorOf(node);
37     next ← map.get(succ);
38     if current is not ⊥ ∧ next is not NULL then
39         gotoStmt ← instruction(Goto(next));
40         insertExpr(body, gotoStmt, "after", current);
41 fun ← fun("dummyMainMethod",args,body);
42 return fun;
```

Algorithm

How to Use

// API Invocation

```
DDMSootConfig.ANDROID_PLATFORM_PATH = $androidPlatformPath$;
DDMSootConfig.ApkPath = $apkPath$;
DDMSootConfig.output = $outputPath$;

DDMSootConfig.sootInitialization();

Set<SootClass> sootClasses = new HashSet<SootClass>(Scene.v().getApplicationClasses());
for (SootClass sootClass : sootClasses) {
    SootMethod sootMethod = DMMFactory.createDDM(sootClass);
        // do your task
}
```

How to Use

// Command Line

```
java -cp DMMPP.jar cn.ac.ios.dmmpp.DMMMain $androidPlatformPath$ $apkPath$ $outputPath$

#output:
$outputPath$ + "DMMPP_"+apkName
```

e.g.

```
java -cp DMMPP.jar cn.ac.ios.dmmpp.DMMMain ./android-platforms ./apks/ch.bailu.aat.apk  ./output

#output:
./output/DMMPP_ch.bailu.aat.apk    // as your input
```

Evaluation: Benefits for Analyzer

App	#C	Explored Paths			Construction Time (ms)		
		FL	D	Δ	FL	D	Δ
app.fedila	6	6	12	**+6**	235	2960	2725
ch.bailu.a	17	17	37	**+20**	414	2,038	1,624
com.asdoi.	6	6	12	**+6**	72	2,262	2,190
com.gimran	13	13	30	**+17**	199	421	222
com.mobile	14	14	153	**+139**	1,111	516	-595
com.tuyafe	5	5	10	**+5**	30	1,724	1,694
com.ubersp	1	1	13	**+12**	40	5	-35
jp.takke.c	7	7	14	**+7**	42	1,932	1,890
net.gitsai	8	8	2,104	**+2,096**	2,775	534	-2,241
xyz.myachi	8	8	1,588	**+1,580**	6,991	2,377	-4,614
appnewness	7	7	35	**+28**	191	554	363
com.alb.pl	1	1	5	**+4**	12	4	-8
com.e.ulil	5	5	10	**+5**	32	460	428
com.pinayr	11	11	293	**+282**	2,094	187	-1,907
f.fajrak.b	9	9	46	**+37**	278	517	239
it.discors	9	9	49	**+40**	306	998	692
kick.wpapp	8	8	40	**+32**	240	486	246
kr.ieodo.a	5	5	10	**+5**	27	530	503
net.easyjo	7	7	14	**+7**	42	1,427	1,385
usd.aleavt	3	3	99	**+96**	459	59	-400
Total	150	150	4,574	**+4,424**	15,590	19,991	4,401

Totally

- Explored Paths: 29.5 times
  29.5 = 4,424/150
- Time Overhead: 4.4s

Each Component

- Explored Paths: +28.5
  28.5 = 4,424/150-1
- Time Overhead: 0.22s

Cause Analysis

// Skip the complete component [1]

```
@Override
protected SootMethod createDummyMainInternal() {
        // Before-class marker
        Stmt beforeComponentStmt = Jimple.v().newNopStmt();
        body.getUnits().add(beforeComponentStmt);

        Stmt endClassStmt = Jimple.v().newNopStmt();
        try {
                // We may skip the complete component
                createIfStmt(endClassStmt);
```

```
void dummyMain(){
    int i = 0;
    if(i == 0){
        return;
    }else{
        lifecycle methods;
    }}
```

[1] https://github.com/secure-software-engineering/FlowDroid/blob/develop/soot-infoflow-android/src/soot/jimple/infoflow/android/entryPointCreators/components/AbstractComponentEntryPointCreator.java#L186

Conclusion

- Constructing the dummy main method with path-sensitive predicates for Android application analysis
- Multiple ways for analyzers to use it
  - API invocation
  - Command line
- Benefiting the path-sensitive analyzer with a very low time overhead

DMMPP

Constructing Dummy Main Methods for Android Apps with Path-sensitive Predicates

Baoquan Cui **Jiwei Yan** **Jian Zhang**

崔保全 燕季薇 张健

Publicly Available at https://github.com/cuixiaoyiyi/DMMPP

Institute of Software, Chinese Academy of Sciences (ISCAS)

中国科学院软件研究所