Automated Testing with Targeted Event Sequence Generation

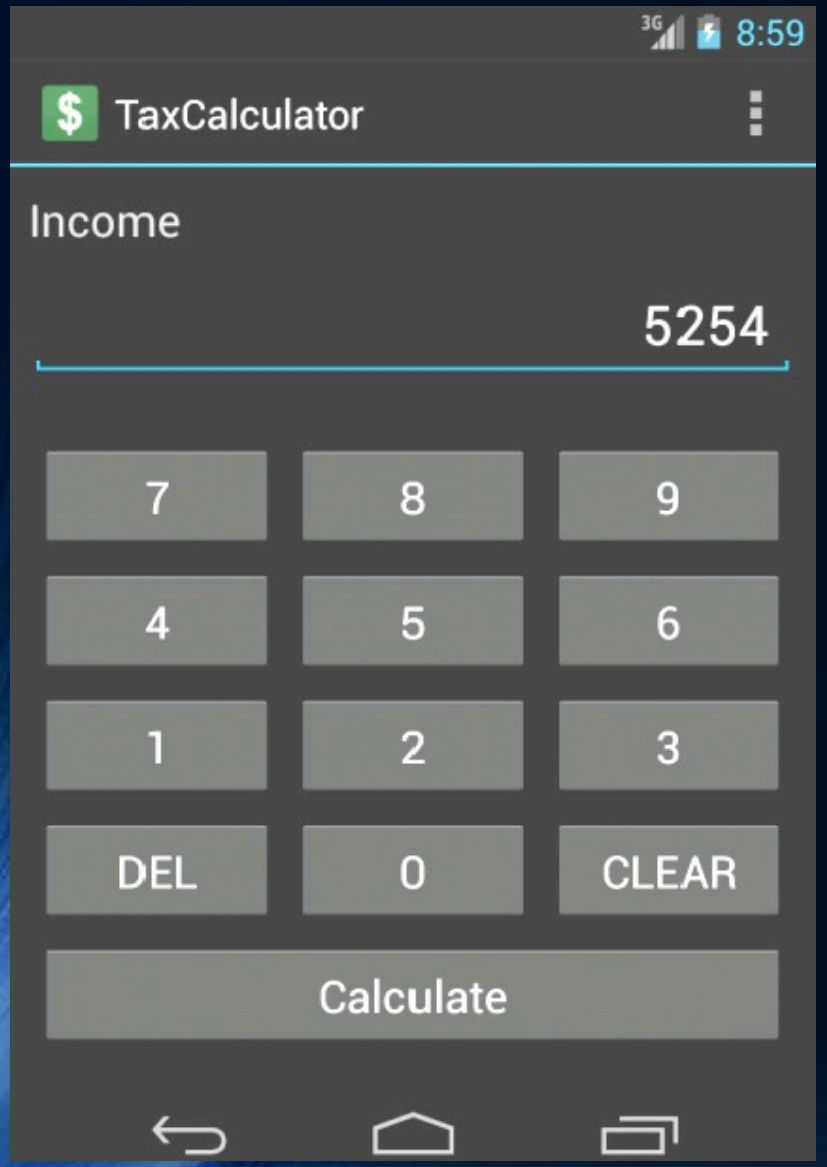
AUTHORS: CASPER S. JENSEN, MUKUL R. PRASAD, ANDERS MØLLER

PRESENTER NAME(S) REMOVED FOR FERPA CONSIDERATIONS

Terminology

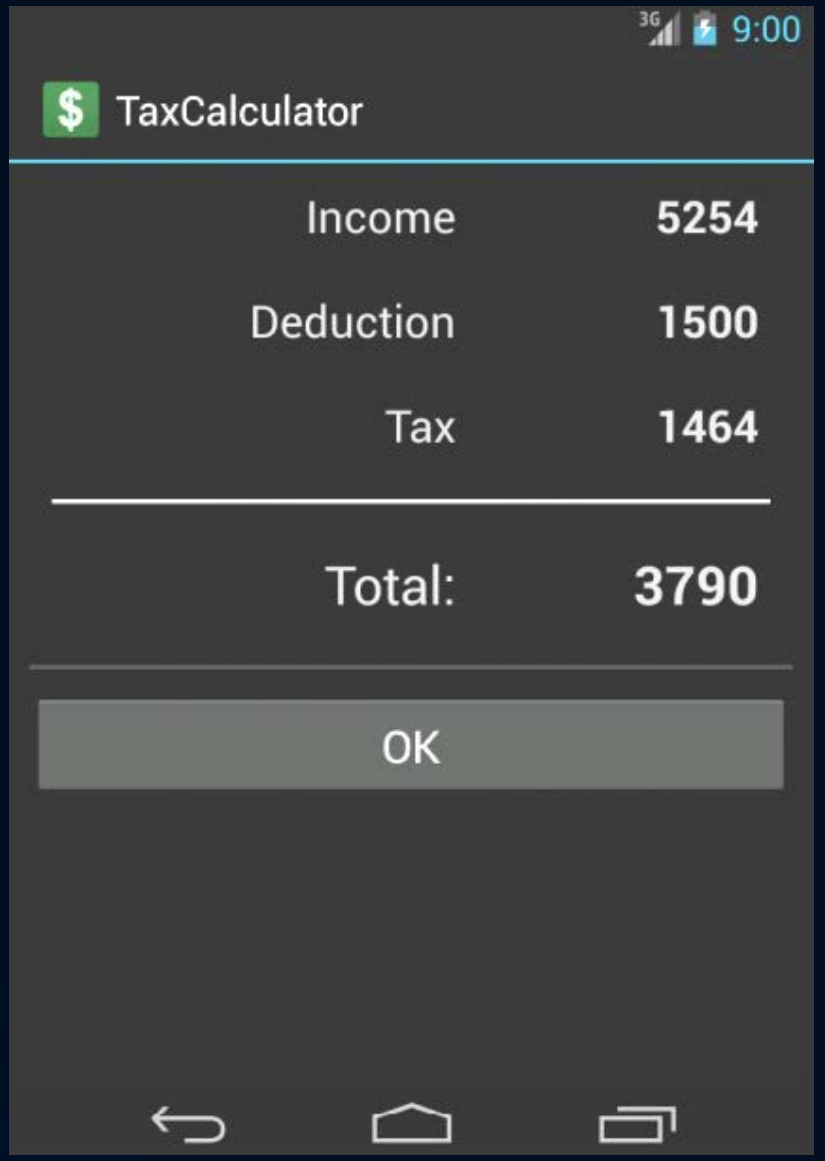
- Anchor Events:
- Connector Events:
- Partial Sequence:

Tax Calculator App



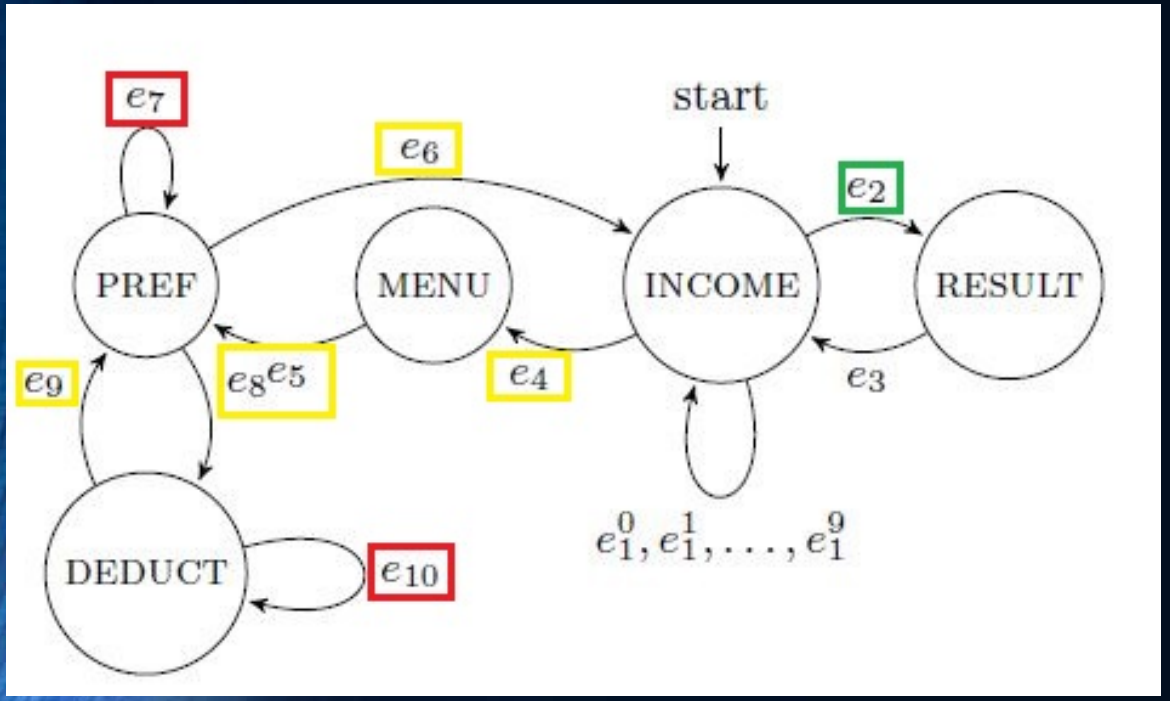
← INCOME screen

RESULT screen →



Example App and UI Model

- TaxCalculator App




```
1  income = this.appState.enteredAmount;
2  deduction = 0;
3  if (Settings.getEnableTaxDeduction()) {
4    deduction = Settings.getTaxDeduction();
5  }
6  taxable = income - deduction;
7  if (taxable < 0) {
8    taxable = 0; // example target
9  }
10 tax = taxable * TAX_RATE;
11 result = income - tax;
```

OnCreate function of the Result Screen

UI Model of TaxCalculator App

M = (S, So, E, T)

Shortest event sequence to target: e4, e5, e7, e8, e10, e9, e6, e2.

Research Questions

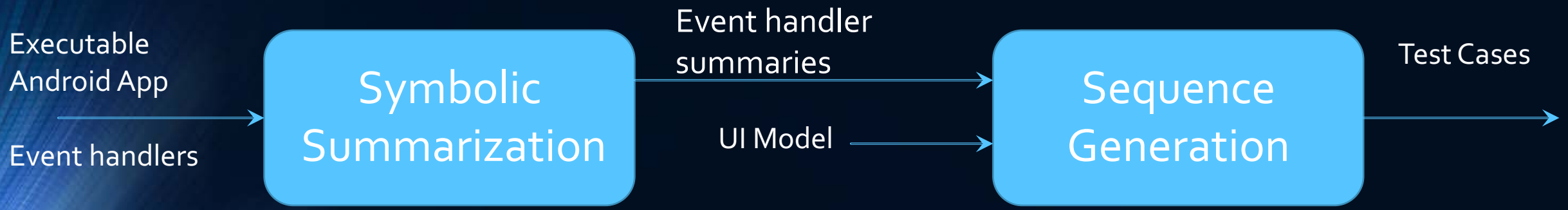
- Is this algorithm able to generate test cases for challenging targets in real-world Android applications?
- Does the use of anchors and connectors have an effect on the ability to reach the targets?
- Do the prioritization heuristics have an effect on the ability to reach the targets?

Inspiration for Key Idea

- Line reachability problem in C
  - Work backward from the target in the call-graph
- Model-based testing technique
  - get event handlers connections and dependencies.

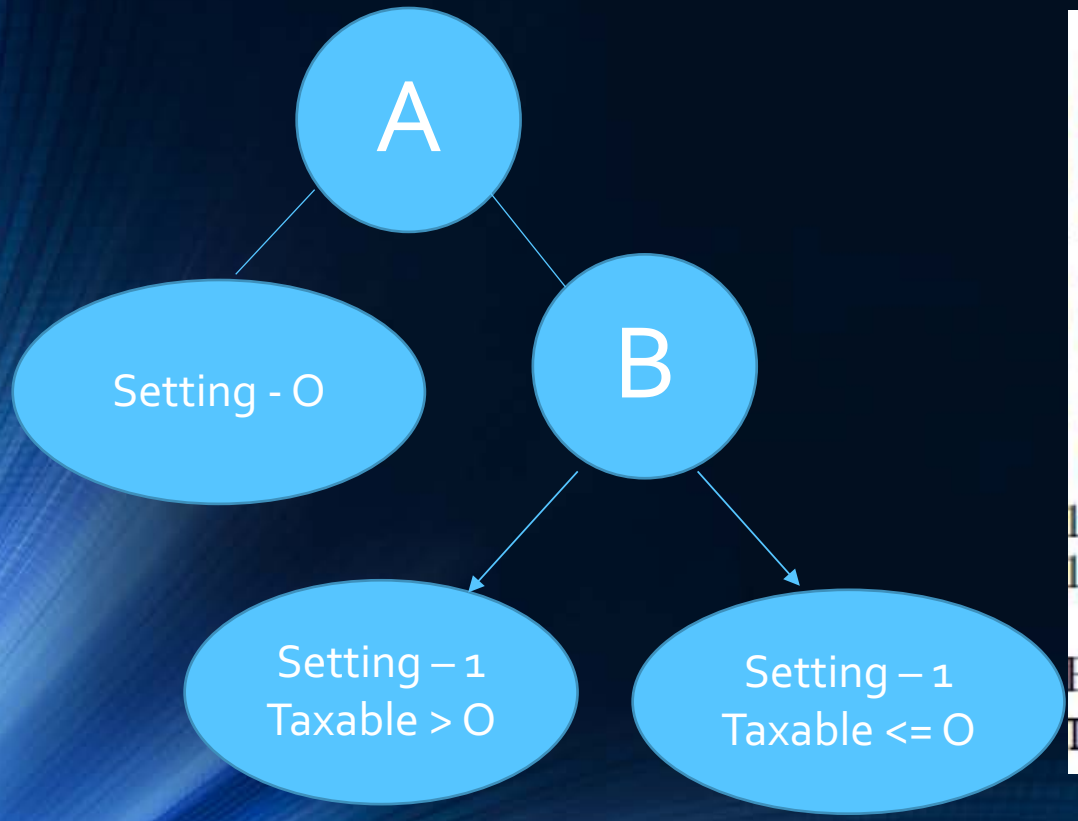
Key Idea

- Symbolic summarization
- Sequence Generation



Symbolic summarization

- locate event handlers using the UI model
- Each iteration of concolic execution explore one path and computes its path summary.




```
1 income = this.appState.enteredAmount;
2 deduction = 0;
3 if (Settings.getEnableTaxDeduction()) {
4   deduction = Settings.getTaxDeduction();
5 }
6 taxable = income - deduction;
7 if (taxable < 0) {
8   taxable = 0; // example target
9 }
10 tax = taxable * TAX_RATE;
11 result = income - tax;
```

Figure 3: Snippet from the `onCreate` method in the `TaxResult` activity in *TaxCalculator*.

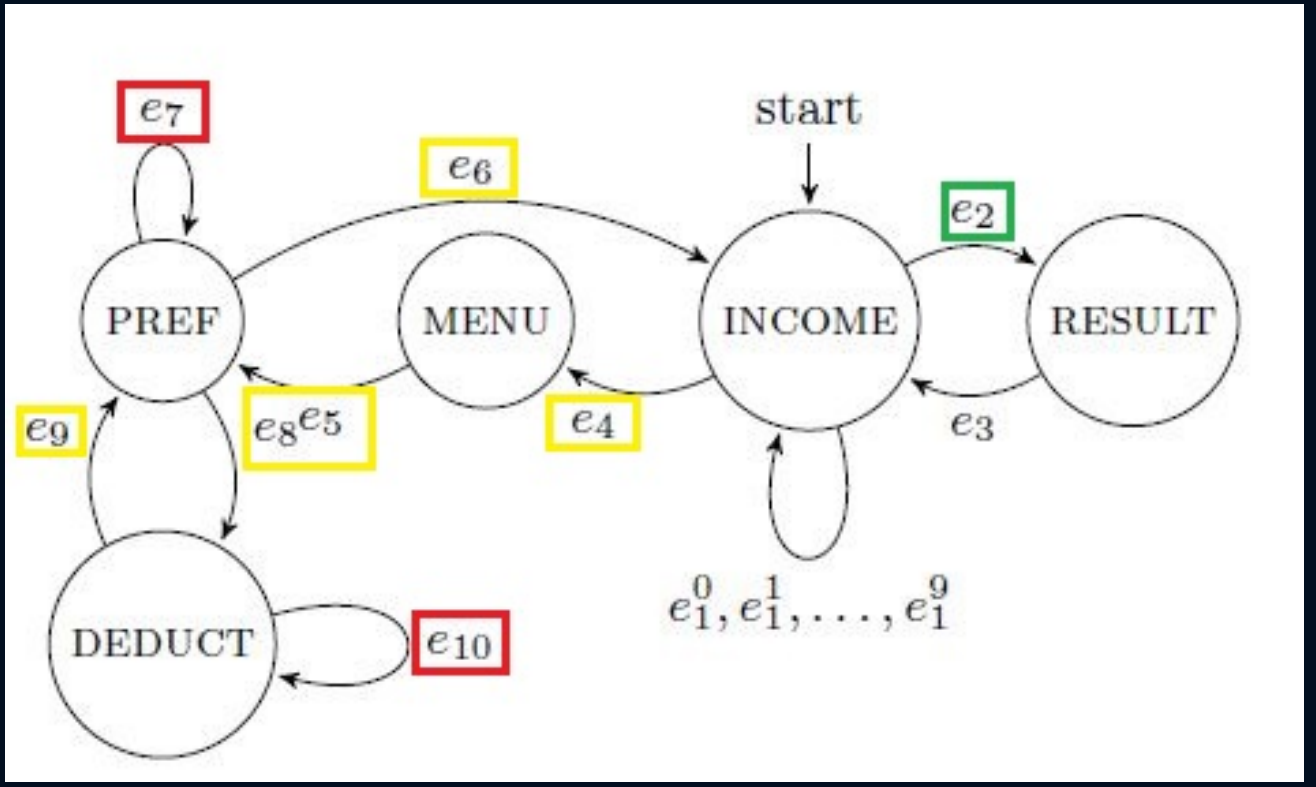
Sequence Generation

- gradually explore sequences of events backward, from the target to the application entry point.

- Initialize the worklist with target event,

  in our example: Initialize by just e2

Construction of Anchors



- Found based on event summaries
  - Breadth First Backward traversal starting from the partial sequence.
  - On each iteration compare dependency set to the event handler of the transition

Construction of Connector Sequences

For each anchor, extract a set of feasible sequences of connectors,

anchor -> partial sequence

- Find all such sequences
- For each check: Path is acyclic + None of labels in the sequence are Anchors
- For each of the shortlisted paths: Check
  if (symbolic state of the anchor == composition of symbolic summaries of the connectors)

Contribution-1

- Combines concolic execution and UI models for targeted generation of test inputs to reach application code that may require many events and with highly constrained event parameters.

Contribution-2

- The framework can be tuned by the prioritization mechanism of the worklist.
  - *Equivalent-Anchors Reprioritization*
  - *Connector Reprioritization*
  - *Increment-Decrement Reprioritization*

Evaluation Summary

- Implementation:
  - Collider (8ooo LOC) without libraries
  - Symbolic summarization and Sequence Generation
- Application criteria
  - Publically available source code for manually UI Model generation
  - UI driven application, but not computationally intensive
  - Contains branches that depend on previous event or event parameters
  - Well representation of the different application categories

Benchmark application

- TippyTipper (1800 LOC): a tip percentage calculator
- ConnectBot (33000 LOC): a SSSH client
- MunchLife (400 LOC): a utility for keeping score in a card game
- OpenManager (2500 LOC): a file manager
- DieDroid (1900 LOC): a virtual dice rolling application

Baseline tools

- Simple Crawler
  - produces event systematically based entirely on the UI models without considering the application code.
- Monkey
  - A random testing tool provided by Android SDK .

Experiment Result

Benchmark	Targets depending on Sequence	Targets depending on Parameters	Reached	Potential	Average size Test case	Average size Connectors	Pruning of anchors
TippyTipper	15	1	7	9	13	9	71%
ConnectBot	17	25	16	26	8	4	58%
Munchlife	5	5	6	4	29	7	66%
OpenManager	11	7	9	9	8	4	39%
DieDroid	2	11	8	5	10	4	38%

- Note that for evaluation of Collider, we only consider the challenging targets
  - all targets of interest depend on the sequencing of events beyond what is expressible in UI models alone.

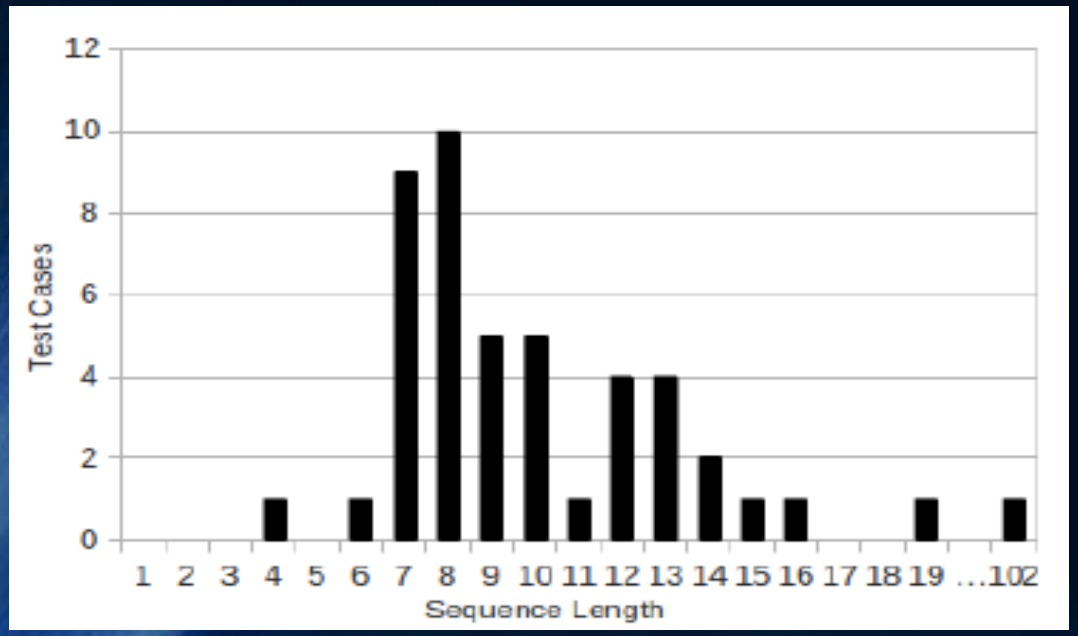
Answers to Research Question -1

Benchmark	Targets depending on Sequence	Targets depending on Parameters	Reached	Potential	Average size Test case	Average size Connectors	Pruning of anchors
TippyTipper	15	1	7	9	13	9	71%
ConnectBot	17	25	16	26	8	4	58%
Munchlife	5	5	6	4	29	7	66%
OpenManager	11	7	9	9	8	4	39%
DieDroid	2	11	8	5	10	4	38%

- Is this algorithm able to generate test cases for challenging targets in real-world Android applications?

Answers to Research Question -2

Benchmark	Targets depending on Sequence	Targets depending on Parameters	Reached	Potential	Average size Test case	Average size Connectors	Pruning of anchors
TippyTipper	15	1	7	9	13	9	71%
ConnectBot	17	25	16	26	8	4	58%
Munchlife	5	5	6	4	29	7	66%
OpenManager	11	7	9	9	8	4	39%
DieDroid	2	11	8	5	10	4	38%



- Does the use of anchors and connectors have an effect on the ability to reach the targets?

Answers to Research Question -3

Benchmark	Targets depending on Sequence	Targets depending on Parameters	Reached	Potential	Average size Test case	Average size Connectors	Pruning of anchors
TippyTipper	15	1	7	9	13	9	71%
ConnectBot	17	25	16	26	8	4	58%
Munchlife	5	5	6	4	29	7	66%
OpenManager	11	7	9	9	8	4	39%
DieDroid	2	11	8	5	10	4	38%

- Do the prioritization heuristics have an effect on the ability to reach the targets?

Discussion Questions -1

Will using Depth First Backward Traversal instead of Breadth First make a difference in identifying Anchor event? If so how?.

Discussion Questions -2

There are only portion of the challenging targets were reached. Why Collider missed those targets?

Discussion Questions -3

When evaluation, authors indicated one preprocess step, generated UI Model, in this case they were manually generated. But for some large applications, how is scaling ability of the program?

Discussion Questions -4

What would happen if we want Collider to generate test cases in simple target?

Discussion Questions -5

Does the technique proposed in the paper work for other type of the event driven application? even in different platform such as IOS or Windows?

Summary

- Automated Test generation for Event driven apps
- By Symbolic Summarization n Sequence Generation
- Prioritization rules to limit the number of candidate sequences
- Collider works well for generating challenging event sequences

Thank You