2025
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 24-27, 2025

Reaching 100% Functional Coverage Using Machine Learning: A Journey of Persistent Efforts

Jaecheon Kim, Taewook Nam, Wonil Cho

SAMSUNG

accellera
SYSTEMS INITIATIVE

Contents

- Background
  - A Key Factor of Functional Coverage Closure
  - Uniform/Non-Uniform Weighted Distribution Experiment
- Proposed Approach
  - Suggested Approach Overview
  - Simple Functional Coverage Example
  - Major Points of the Proposed Approach with Functional Coverage Example
- Experimental Result
- Conclusion

A Key Factor of Functional Coverage Closure

- Cross coverage involves previously declared coverage points
- Verifying if the behavior of a specific combination of the device under DUT
- One of the key challenges in achieving functional coverage closure

```
constraint test_cr {
   test_a dist {0:=5,1:=5};   // 2 cases
   test_b dist {[2:7]};       // 6 cases
   test_c dist {[8:12]};      // 5 cases
   test_d dist {[13:19]};     // 7 cases
}
```

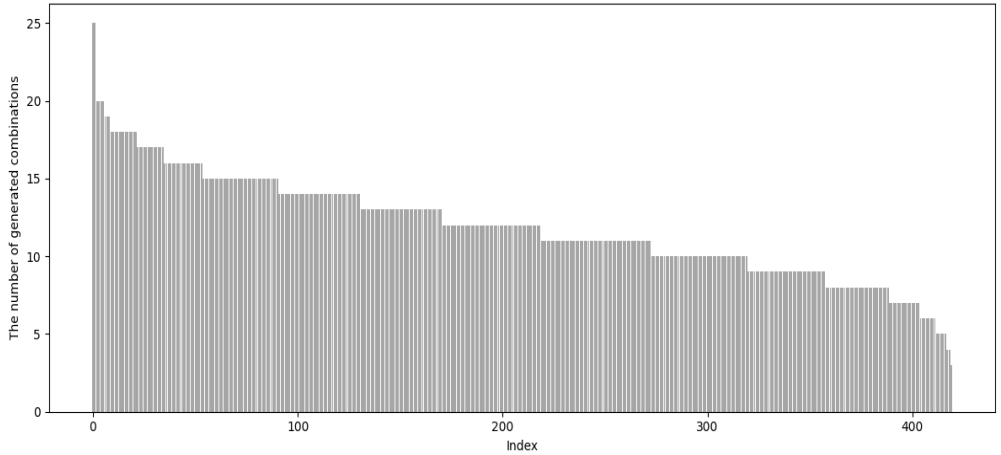
```
covergroup cg_test;
   option.per_instance = 1;
   option.name = "test_cov";

   cp_test_a  : coverpoint test_a { bins b[] = {[0:1]}; }
   cp_test_b  : coverpoint test_b { bins b[] = {[2:7]}; }
   cp_test_c  : coverpoint test_c { bins b[] = {[8:12]}; }
   cp_test_d  : coverpoint test_d { bins b[] = {[13:19]}; }

   cross_test_abcd  : cross cp_test_a, cp_test_b, cp_test_c, cp_test_d;
endgroup
```

Uniform Weighted Distribution Experiment

- The cross coverage, cross_test_abcd, consists of a total of 420 bins.
- Results of 5,000 randomizations when the weighted distribution is uniform
- Difference between the MAX (25) and the MIN (3) of the frequency distribution

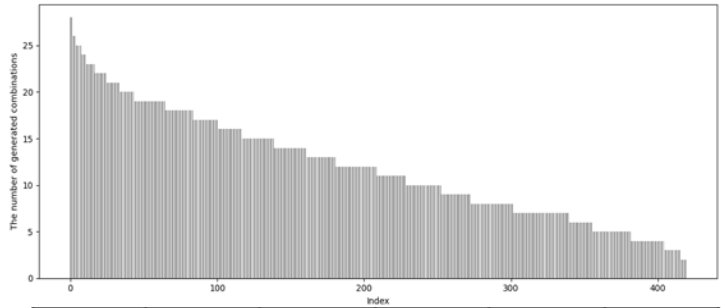


index	test_a	test_b	test_c	test_d	count
0	1	2	12	14	25
1	0	4	12	15	25
2	0	4	9	19	20
417	0	7	8	18	4
418	1	2	8	14	4
419	0	3	10	18	3

Non-Uniform Weighted Distribution Experiment

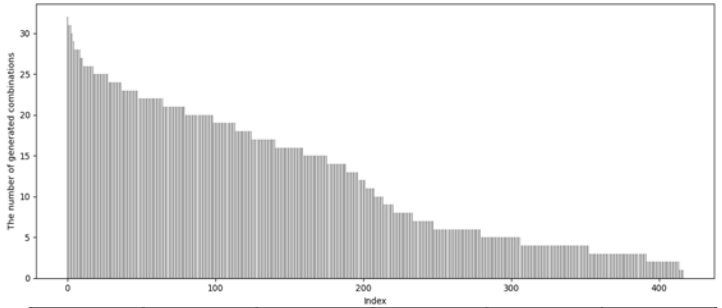
- Results of 5,000 randomizations when the weighted distribution is non-uniform
- Certain combinations are no longer generated

test_a dist {0:=3,1:=7};



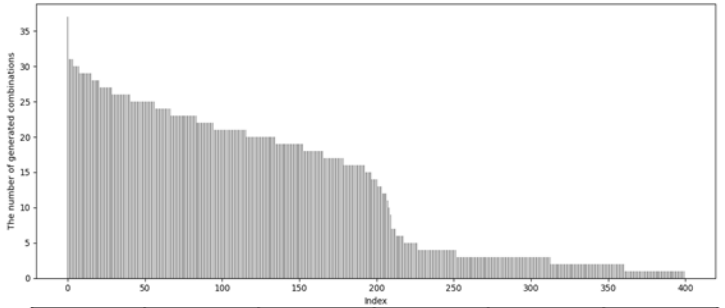
index	test_a	test_b	test_c	test_d	count
0	1	7	10	19	28
1	1	4	10	13	28
2	1	5	8	13	26
417	0	3	10	14	2
418	0	5	12	17	2
419	0	6	9	18	2

test_a dist {0:=2,1:=8};



index	test_a	test_b	test_c	test_d	count
0	1	3	9	13	32
1	1	6	9	15	31
2	1	5	10	15	31
417	0	5	8	15	0
418	0	7	10	17	0
419	0	7	12	14	0

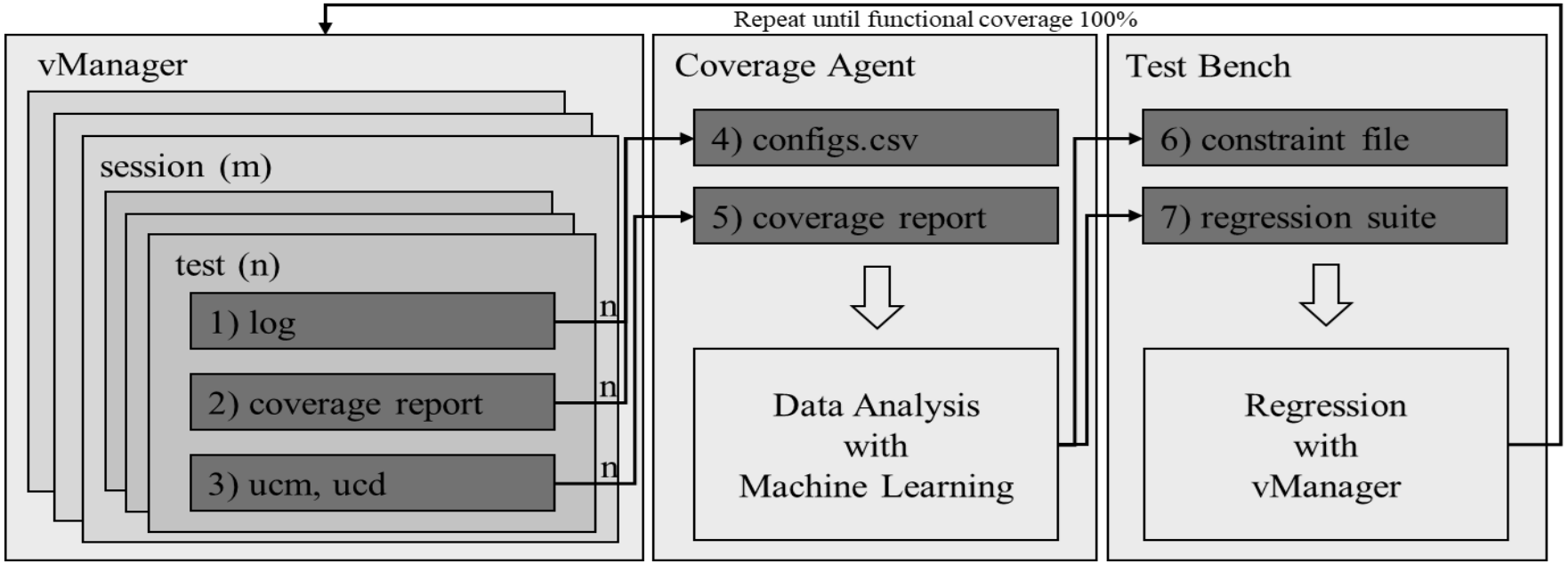
test_a dist {0:=1,1:=9};



index	test_a	test_b	test_c	test_d	count
0	1	5	11	16	37
1	1	2	10	19	31
2	1	7	9	13	31
417	0	7	9	17	0
418	0	7	9	19	0
419	0	7	11	18	0

Suggested Approach Overview

- Python-based coverage agent that supports this workflow
- Automated functional coverage closure using machine learning



Simple Functional Coverage Example(1)

- Total of 8 coverpoints and 2 cross coverages declared
- cross_test_mode has a uniform distribution
- cross_test_xy has a non-uniform distribution

```
constraint test_cr {
   test_on inside {0,1};
   test_x dist{0:=1, [1:20]:=3, [21:30]:=3, [31:40]:=2, 41:=1};
   test_y dist{0:=1, [1:20]:=4, [21:30]:=4, 31:=1};

   test_mode0 inside {[0:8]};
   test_mode1 inside {[0:3]};
   test_en inside {[0:1]};
   test_bypass inside {[0:1]};
   test_mux inside {[0:4]};

   dummy0 inside {'h20,'h21,'h22,'h23,'h24};
   dummy1 inside {'h0,'h1,'h2,'h3};
   dummy2 inside {'h4,'h5,'h6,'h7};
   dummy3 inside {'h8,'h9,'hA,'hB};
   dummy4 inside {'hc,'hd,'hE,'hF};
}
```

```
covergroup cg_test;
   option.per_instance = 1;
   option.name = "test_cov";

   cp_test_on       : coverpoint test_on  { bins b[]  = {[0:1]};   }
   cp_test_x        : coverpoint test_x   { bins b[]  = {[0:41]}; }
   cp_test_y        : coverpoint test_y   { bins b[]  = {[0:31]}; }

   cp_test_en       : coverpoint test_en       { bins b[]  = {[0:1]}; }
   cp_test_mux      : coverpoint test_mux      { bins b[]  = {[0:4]}; }
   cp_test_mode0    : coverpoint test_mode0    { bins b[]  = {[0:8]}; }
   cp_test_mode1    : coverpoint test_mode1    { bins b[]  = {[0:3]}; }
   cp_test_bypass   : coverpoint test_bypass   { bins b[]  = {[0:1]}; }

   cross_test_xy    : cross cp_test_on, cp_test_x, cp_test_y;
   cross_test_mode : cross cp_test_en, cp_test_mux, cp_test_mode0, cp_test_mode1, cp_test_bypass;
endgroup
```

Simple Functional Coverage Example(2)

- The coverage results after running the example code with regression

The accumulated number of tests	cross_test_xy		cross_test_mode	
	Coverage rate (%)	Touched bins / Total bins	Coverage rate (%)	Touched bins / Total bins
1000	30.80	828 / 2688	75.42	543 / 720
2000	50.74	1364 / 2688	94.31	679 / 720
3000	65.36	1757 / 2688	98.33	708 / 720
4000	75.33	2025 / 2688	99.03	713 / 720
5000	81.58	2193 / 2688	99.72	718 / 720
6000	85.86%	2308 / 2688	99.86	719 / 720
7000	88.99%	2392 / 2688	100.00	720 / 720
8000	91.11%	2449 / 2688	100.00	720 / 720
9000	92.67%	2491 / 2688	100.00	720 / 720
10000	94.12%	2530 / 2688	100.00	720 / 720

(a)

Coverpoint name	Coverage rate (%)	Touched bins / Total bins
cp_test_on	100.00	2 / 2
cp_test_x	100.00	42 / 42
cp_test_y	100.00	32 / 32
cp_test_mode0	100.00	9 / 9
cp_test_mode1	100.00	4 / 4
cp_test_en	100.00	2 / 2
cp_test_bypass	100.00	2 / 2
cp_test_mux	100.00	5 / 5

(b)

Log Parsing of Random Variables

- UVM code snippets for printing random variables in the

```
`uvm_object_utils_begin(test_cfg_c)
    `uvm_field_int(test_on, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(test_x, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(test_y, UVM_ALL_ON + UVM_DEC)

    `uvm_field_int(test_en, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(test_mux, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(test_mode0, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(test_mode1, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(test_bypass, UVM_ALL_ON + UVM_DEC)

    `uvm_field_int(dummy0, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(dummy1, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(dummy2, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(dummy3, UVM_ALL_ON + UVM_DEC)
    `uvm_field_int(dummy4, UVM_ALL_ON + UVM_DEC)
  `uvm_object_utils_end
```

(a)

```
uvm_default_printer = uvm_default_line_printer;
uvm_default_printer.knobs.begin_elements=-1;
```

(b)

Coverage Report Example

- What we can get:
  - Name of cross coverage
  - Total number of bins
  - Current number of touched bins
  - Number of coverpoints that make up the cross coverage

test_cov	21.18%, 0.29% (10/3506)	36 (test_cov.sv) covergroup cg_test;
|--cross_test_xy	0.04% (1/2688)	50 (test_cov.sv) cross_test_xy : cross cp_test_on, cp_test_x, cp_test_y;
| |--b[0],b[0],b[0]	100.00% (1/1)	50 (test_cov.sv) cross_test_xy : cross cp_test_on, cp_test_x, cp_test_y;
| |--b[0],b[0],b[1]	0.00% (0/1)	50 (test_cov.sv) cross_test_xy : cross cp_test_on, cp_test_x, cp_test_y;
| |--b[0],b[0],b[2]	0.00% (0/1)	50 (test_cov.sv) cross_test_xy : cross cp_test_on, cp_test_x, cp_test_y;
| |--b[0],b[0],b[3]	0.00% (0/1)	50 (test_cov.sv) cross_test_xy : cross cp_test_on, cp_test_x, cp_test_y;
| |--b[0],b[0],b[4]	0.00% (0/1)	50 (test_cov.sv) cross_test_xy : cross cp_test_on, cp_test_x, cp_test_y;

The Use of Machine Learning

- Decision Tree Algorithm is used
  - Input features : Random variables
  - Target : Cross coverage numbering
- Predicting highly correlated random variables

cross_test_xy	numbering	cross_test_mode	numbering
b[0],b[0],b[0]	1	b[0],b[0],b[0],b[0],b[0]	1
b[0],b[0],b[1]	2	b[0],b[0],b[0],b[0],b[1]	2
b[0],b[0],b[2]	3	b[0],b[0],b[0],b[0],b[2]	3
b[1],b[41],b[29]	2677	b[8],b[3],b[1],b[1],b[2]	718
b[1],b[41],b[30]	2678	b[8],b[3],b[1],b[1],b[3]	719
b[1],b[41],b[31]	2688	b[8],b[3],b[1],b[1],b[4]	720

The Machine Learning Prediction Results

- cross_test_xy : Find 3 random variables out of the 13
- cross_test_mode : Find 5 random variables out of the 13

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
cross_test_xy	test_cfg.test_x	test_cfg.test_y	test_cfg.test_y	test_cfg.test_y	test_cfg.test_x
	test_cfg.test_y	test_cfg.test_x	test_cfg.test_x	test_cfg.test_x	test_cfg.test_y
	test_cfg.dummy1	test_cfg.test_mode0	test_cfg.test_on	test_cfg.test_mode0	test_cfg.test_on
cross_test_mode	test_cfg.test_mode0	test_cfg.test_mux			
	test_cfg.test_mux	test_cfg.test_mode0			
	test_cfg.test_x	test_cfg.test_en			
	test_cfg.test_y	test_cfg.test_bypass			
	test_cfg.test_en	test_cfg.test_mode1			

Post-processing of Machine Learning Results

- Random variable product set for cross coverage (cross_test_xy)

Index	test_cfg.test_y	test_cfg.test_x	test_cfg.test_on	Count
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
2685	31	40	1	1
2686	31	41	0	0
2687	31	41	1	0

Constraint File Example

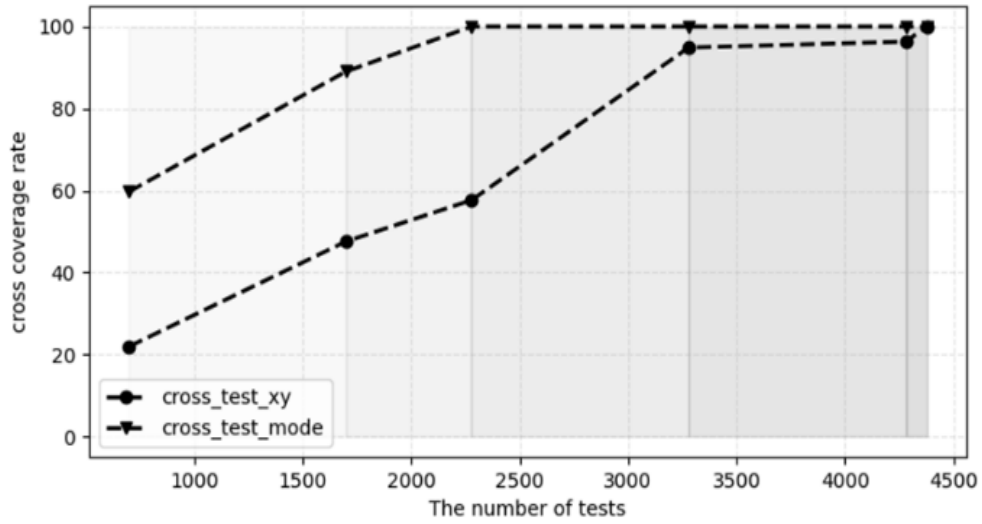
- The randomize() with constraint block.
- Random variables are declared as "soft“

```
//for index 0 in the table on the previous slide
soft test_cfg.test_y == 0;  soft test_cfg.test_x == 0;  soft test_cfg.test_on == 0;

//for index 2687  in the table  on the previous slide
soft test_cfg.test_y == 31; soft test_cfg.test_x == 41; soft test_cfg.test_on == 1;
```

Simple Example Code Result

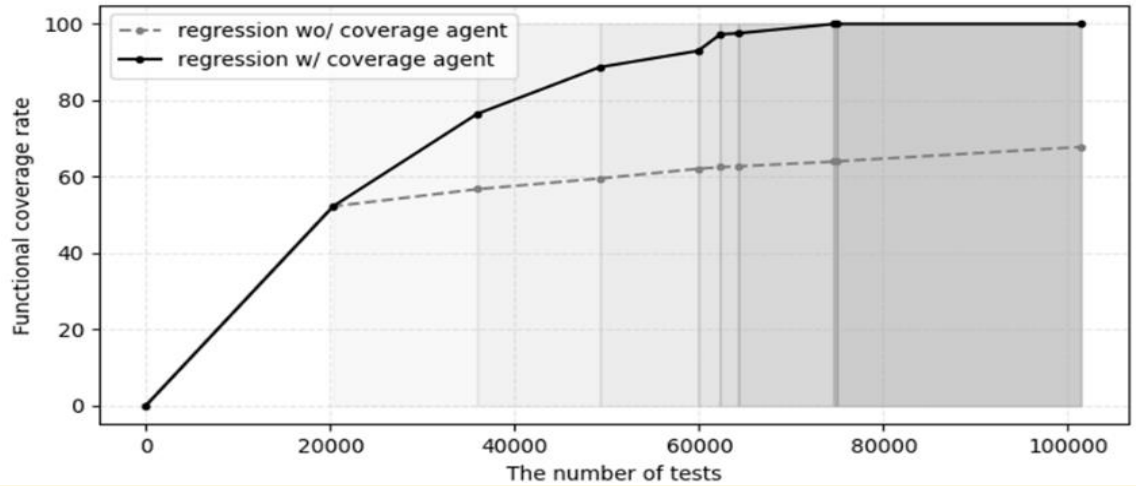
- The results of applying the coverage agent to the example code
- Maximum of 1,000 tests per iteration



Iteration	The accumulated number of tests	Number of tests by coverage agent	cross_text_xy (%)	cross_test_mode (%)
0	700	-	21.95	59.72
1	1700	1000	47.58	89.03
2	2279	579	57.66	100
3	3279	1000	94.87	100
4	4279	1000	96.32	100
5	4378	99	100	100

Experimental Result (1)

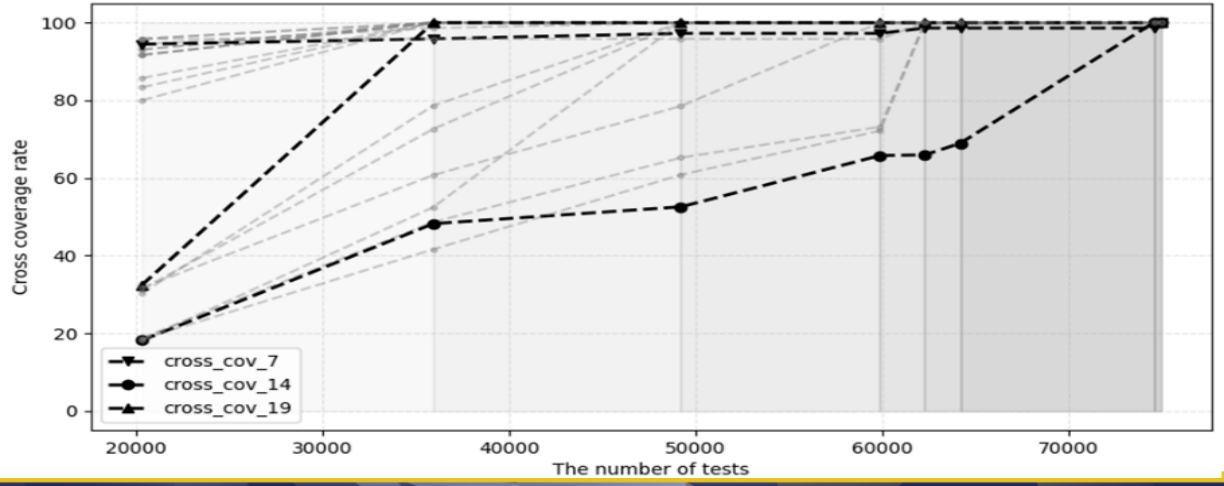
- Functional coverage closure with coverage agent significantly faster than the baseline.
- When the previous regression reached 64.98%, the regression run with the coverage agent reached 100%.



The accumulated number of tests	Number of tests by coverage agent	wo/ coverage agent (%)	w/ coverage agent(%)
20290	-	52.17	52.17
35929	15639	56.68	76.43
49210	13281	59.48	88.66
59896	10686	62.03	92.94
62301	2405	62.45	97.29
64189	1888	62.66	97.53
74585	10396	63.90	99.99
74956	371	63.98	100.00
101450	-	67.74	100.00

Experimental Result (2)

- Until it is completely filled
  - cross_cov_7 : Almost not filled or not filled at all.
  - cross_cov_14 : Gradually filled up.
  - cross_cov_19 : Filled up immediately in the next regression.



The accumulated number of tests	cross_cov _7(%)	cross_cov _14(%)	cross_cov _19(%)
20290	94.44	18.17	32.42
35929	95.83	48.23	100
49210	97.22	52.56	100
59896	97.22	65.77	100
62301	98.61	65.93	100
64189	98.61	68.92	100
74585	98.61	100	100
74956	100	100	100

Conclusion

- No need for DV engineers to manually analyze the holes and modify the constraints of random variables.
- Functional coverage closure can be achieved markedly faster than the original regression without any additional effort from DV engineers.

Questions

- Thank you