

Efficient Forward Computation of Dynamic Slices Using Reduced Ordered Binary Decision Diagrams

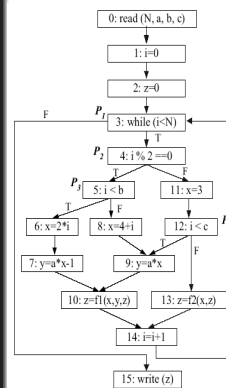
Xiangyu Zhang, Rajiv Gupta, and Youtao Zhang

Outline

- Background
- Slicing Algorithms
- Characteristics of Dynamic Slices
- Efficient Forward Computation
- Results
- Conclusions

Background: Program Slicing

- Introduced by Mark Weiser in 1982
- Objective is to find the relevant subset of program statements
- Static Slicing
 - Large slices
- Dynamic Slicing
 - Smaller slices



- Dynamic slice: $\langle v, i_j \rangle$
 - Variable v
 - Execution point i_j

- Example
 - $\langle z, 15_1 \rangle$
 - $\{0, 1, 2, 3, 4, 11, 12, 13, 15\}$

Slicing Algorithms

- Backward Computation
 - Dependencies saved in a dynamic dependence graph
 - Slices constructed upon request
 - Allows computation of all dynamic slices
 - Uses up all of a machine's memory
 - LP algorithm

Slicing Algorithms

- Forward Computation
 - Slices are maintained as a program executes
 - Only the latest dynamic slices are maintained
 - Significant time cost
- This paper presents an efficient algorithm for forward computation of dynamic slices.

Forward Computation

- $|V|$: number of variables
- $|SE|$: number of statements executed
- $|DSE|$: number of statically distinct statements
- Space Cost
 - $O(|SE| \times |DSE|)$ if all slices are saved
 - $O(|V| \times |DSE|)$ if only the latest slices are saved
- Time Cost
 - $O(|SE| \times |DSE|)$ in the worst case

7

Benchmark Programs

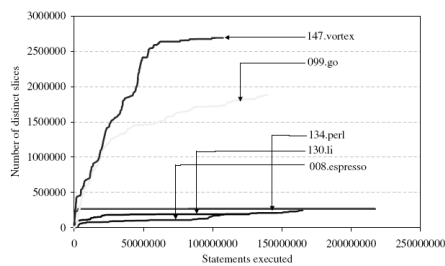
Program	Lines of C Code	$ V $	$ DSE $	$ SE $ (millions)
008.espresso	14,850	43535	22470	165.1
130.li	7,741	156102	10399	126.6
134.perl	27,044	7040	21767	217.5
099.go	29,629	91574	50371	139.5
147.vortex	67,213	65285	69249	107.5
Average	29,295	72707	34851	151.2

- The cost of forward computation of dynamic slices is high!

8

Characteristics of Dynamic Slices

- Reappearing Slices



9

Characteristics of Dynamic Slices

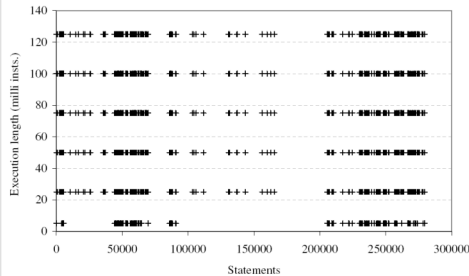
- Overlapping

Program	All Points	Latest Point
008.espresso	26.5%	41.8%
130.li	33.2%	52.6%
134.perl	30.4%	35.0%
099.go	43.0%	16.2%
147.vortex	33.6%	52.6%
Average	33.3%	39.6%

10

Characteristics of Dynamic Slices

- Clustering

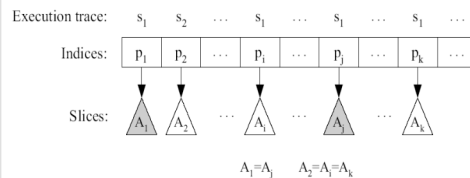


11

Efficient Forward Computation

- First, exploit reappearing slices to efficiently save all dynamic slices

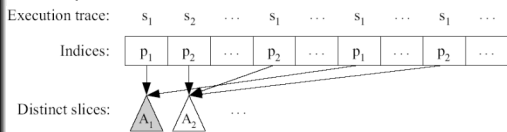
- Step 1



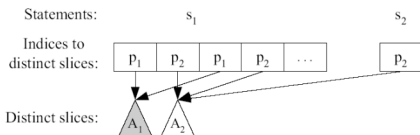
12

Efficient Forward Computation

• Step 2



• Step 3

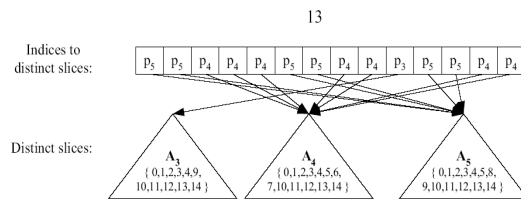


13

SEQUITUR Algorithm

Execution Trace:

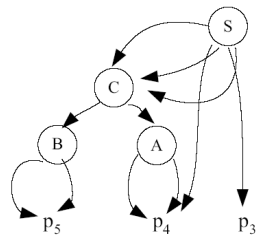
0 1 2 3 4 5 8 9 10 14 3 4 11 12 13 14 3 4 11 12 13 14 3 4 6 7 10 14 3 4 11 12 13 14
 3 4 11 12 13 14 3 4 11 12 13 14 3 4 5 8 9 10 14 3 4 11 12 13 14 3 4 11 12 13 14 3 4 6 7 10 14
 3 4 11 12 13 14 3 4 11 12 13 14 3 4 11 12 9 10 14 3 4 11 12 13 14 3 4 5 8 9 10 14
 3 4 11 12 13 14 3 4 11 12 13 14 3 4 6 7 10 14 3 4 11 12 13 14 3 4 11 12 13 14 3 15



14

SEQUITUR Algorithm

$S \rightarrow C p_4 C p_3 C$
 $C \rightarrow B A$
 $B \rightarrow p_5 p_3$
 $A \rightarrow p_4 p_4$



String = $\{ p_5, p_5, p_4, p_4, p_4, p_5, p_5, p_4, p_4, p_3, p_5, p_5, p_4, p_4 \}$

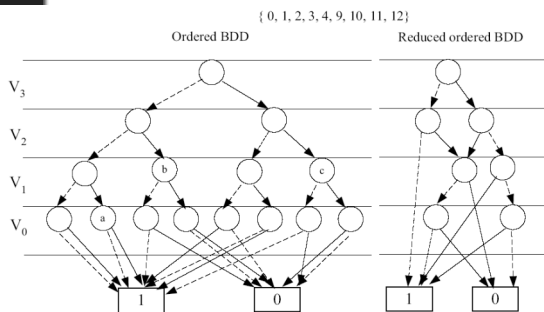
15

Efficient Forward Computation

- Compress distinct dynamic slices based on overlapping and clustering
- Reduced Ordered Binary Decision Diagrams (roBDDs)
 - Sinks
 - 1-edges and 0-edges

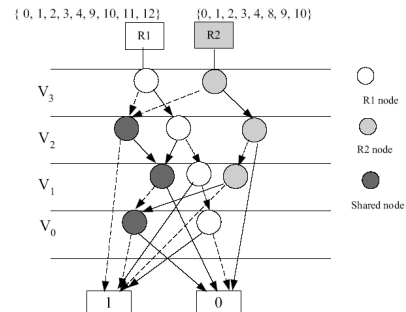
16

Example roBDD



17

roBDDs and Sharing



18

Results

- Algorithms studied
 - Non-BDD
 - roBDD
 - LP
 - oFP
- Benchmarks: 008.espresso, 130.li, 134.perl, 099.go, 147.vortex

19

Non-BDD vs. roBDD: Latest Slice

- Comparison of memory consumption

Program	Space Required		Non-BDD roBDD
	Non-BDD (MB)	roBDD (MB)	
008.espresso	60.0	44.1	1.36
130.li	41.9	28.0	1.50
134.perl	71.7	48.7	1.47
099.go	150.6	100.0	1.51
147.vortex	1084.4	100.0	10.8

20

Non-BDD vs. roBDD: All Slices

- Comparison of memory consumption

Program	Space Required		Non-BDD roBDD
	Non-BDD Estimate (MB)	roBDD (MB)	
008.espresso	5667.8	44.1	128.44
130.li	1753.0	28.0	62.51
134.perl	5192.7	48.7	106.69
099.go	116788.7	255.8	456.54
147.vortex	199713.9	392.8	508.42

21

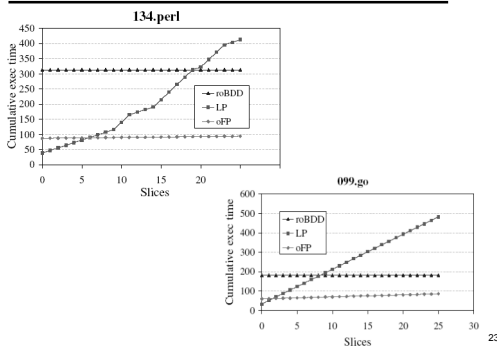
Non-BDD vs. roBDD: Latest Slice

- Comparison of execution times

Program	$ SE $ (millions)	Non-BDD (min.)	roBDD (min.)	Speedup
008.espresso	21.4	487.82	7.74	63
130.li	17.5	544.06	7.26	75
134.perl	30.6	803.81	24.55	33
099.go	20.9	742.30	10.29	72
147.vortex	15.3	990.10	9.98	99

22

roBDD vs. LP and oFP



23

Conclusions

- Identification of statistical characteristics of dynamic slices
- Space and time efficient forward computation algorithm
- Maintenance of all dynamic slices for an execution
- Comparison of backward and forward computation

24