Program Analysis Too Loopy? Set the Loops Aside

Eric Larson

September 25, 2011

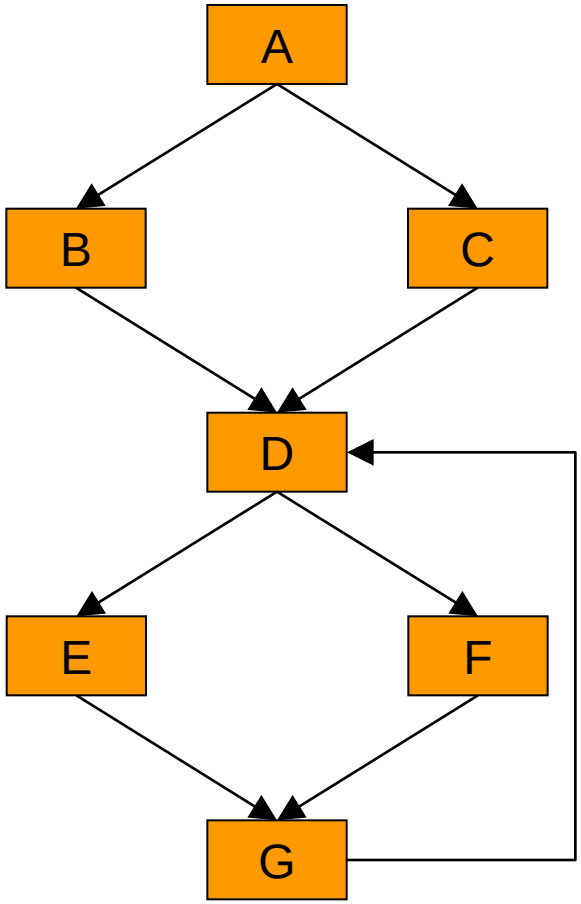
Seattle University

Loops are Important!

- Virtually every program has loops.

Path Based Analysis

- Many bug detection / testing techniques involve traversing different paths within a unit (often a function).
  - Symbolic execution
  - Test generation
- A key issue: Loops introduce a large, possibly infinite, number of paths.



How to Handle Loops?

- Restrict the number of iterations in the loop to some small subset.
  - May miss bugs that only surface on a particular number of iterations.
- Restrict the analysis time: stop analysis after *n* paths or *x* seconds.
  - The traversal order of paths is important to make sure all regions of the code are exercised.
- Compute loop post-conditions.
- Detect and handle loops that meet particular pattern(s).

Interprocedural Summaries

- PREfix* uses symbolic execution to find common C/C++ coding mistakes.
- For function calls:
  - The called function is analyzed first.
  - A model or summary is created that captures the functionality of the called function.
  - The model replaces the function call eliminating the need for interprocedural path analysis.
- Can a similar approach be used for loops?

*Bush, Pincus, and Sielaff. A static analyzer for finding dynamic programming errors. Software – Practice and Experience. 2000.

Research Study Explores …

- Explores:
  - The number of paths in each loop.
  - The number of paths if loops are analyzed separately.
  - An analysis of how often loops contained certain constructs or properties.
- Analyzed 15 different C programs containing 1,091 loops.
- Implemented using an extension to GrammaTech's CodeSurfer.

Number of Paths in Each Loop

Program	Most Paths in Loop	Number of Paths					
		1	2-10	11-100	101-1000	1001 -10k	> 10k
bc	119	54	43	5	1	0	0
betaftpd	37	5	10	2	0	0	0
diff3	581	29	17	4	3	0	0
find	396	20	24	3	3	0	0
flex	464	70	74	6	3	0	0
ft	4	12	11	0	0	0	0
ghttpd	23	11	9	2	0	0	0
gzip	198	87	82	11	1	0	0
indent	30,352,140	58	37	9	2	1	2
ks	36	19	14	2	0	0	0
othello	121	18	7	0	1	0	0
space	164	23	24	4	1	0	0
sudoku	10,368	19	31	7	1	0	1
thttpd	5,768	30	44	5	4	2	0
yacr2	392	46	65	8	4	0	0
TOTAL	30,352,140	501	492	68	24	3	3
		45.9%	45.1%	6.2%	2.2%	0.3%	0.3%

Analyzing Loops Separately

Program	*Paths (loops traversed at most once)*	*Paths (loops analyzed separately)*		
		Total	*Outside Loops*	*Inside Loops*
bc	949,346	56,965	56,487	478
betaftpd	45,692	42,315	42,209	106
diff3	572,718	40,966	39,735	1,231
find	1,966,770	1,804,370	1,803,439	931
flex	7.40E+11	7.22E+11	7.22E+11	1,398
ft	10,594	526	481	45
ghttpd	9,679	1,156	1,075	81
gzip	3.05E+10	2.37E+09	2.37E+09	873
indent	9.82E+17	8.38E+11	8.38E+11	30,421,708
ks	24,452	153	47	106
othello	13,382	13,201	13,034	167
space	6,227	2,011	1,676	335
sudoku	1.94E+09	21,216	10,099	11,117
thttpd	2.84E+12	3.48E+10	3.48E+10	10,345
yacr2	2,249,048	3,104	1,575	1,529

Loop Characteristics

- Loop breakdown:
  - Array traversals 48.7%
  - Data structure traversals 14.8%
  - Sentinel loops 4.5%
  - Input sentinel loops 2.7%
  - Other 29.3%
- Hard to analyze constructs:
  - 24.2% of loops contained an alternate exit (beyond the normal stopping condition).
  - 57.6% of loops contained a function call.

Conclusion

- Most loops have very few paths.
- Separating loops from complex functions does not reduce complexity.
  - Most complex functions have complexity both inside and outside loops.
- Future Work:
  - Analyze loops in different languages such as C++ or Java.
  - Implement symbolic execution where loops are analyzed separately.

Questions?