Interactive Abstract Interpretation

Julian Erhard

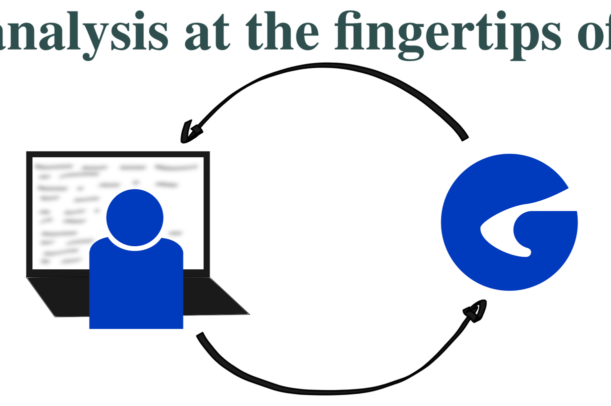
julian.erhard@tum.de

Supervisors: Helmut Seidl & Dirk Beyer

Collaborators: Michael Schwarz (ConVeY) &
Sarah Tilscher (ConVeY) & Simmo Saan (UTartu) & Vesal Vojdani (UTartu)

Goal

Putting static analysis at the fingertips of the developer



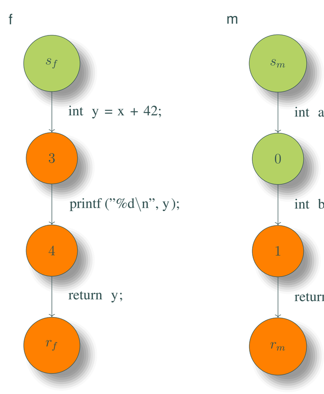
Yard-Sticks

1. Response time:
   Time it takes for the analysis to finish after a program change
2. Consistency:
   Level of precision that is retained compared to a from-scratch analysis
3. Usability:
   Level of integration into the developer workflow

Response-Time

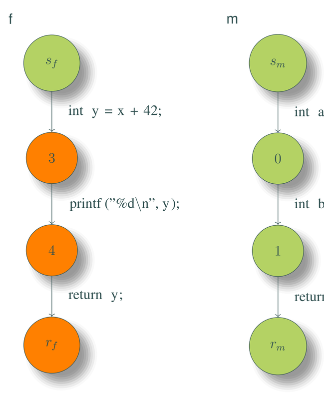
Incremental analysis [2]:

- Exploit dependencies tracked by solver
- Reuse analysis results where possible
- Detect changed functions $F_{changed}$
- Mark results influenced by $f \in F_{changed}$ as unstable
- Restart analysis from return-node r_{main} of `main`



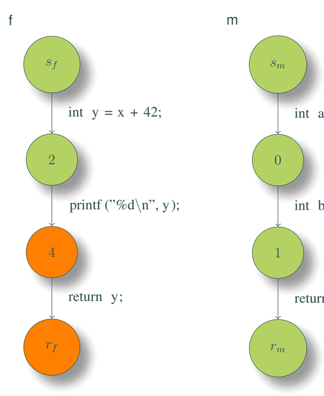
Reluctant destabilization:

- First reanalyze $f \in F_{changed}$
- Only destabilize call-sites of f if results for node r_f changed



Fine-grained change detection:

- Match control-flow-graphs of $f \in F_{changed}$ with previous version
- Reuse results for nodes within f that
  - can be matched, and
  - do not have any new (indirect) predecessor



Incremental postsolver:

- Track unknowns not touched by reanalysis
- Reuse warnings for such unknowns that are still live

```
int f(int x){                    int m(){
        int y = x + 42;                  int a = 25;
        // Add printf                    int b = f(a);
        printf("%d\n", y);               return b;
        return y;                }
}
```

Listing 1: Code example with change. A `printf` is added to `f`.

Consistency

Low precision loss through incremental analysis out-of-the-box.
Issue: Values of flow-insensitive unknowns accumulate over reanalyses.
Solution: Restart subset G of globals, as follows:

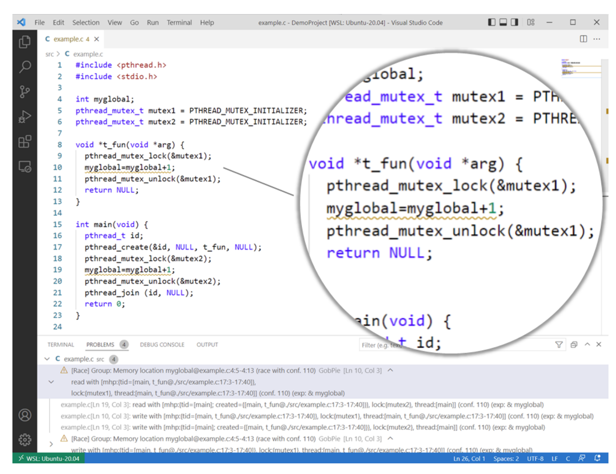
- Reset $g \in G$ to $\bot$
- Set unknowns that side-effected to g and all that (transitively) depend on them to unstable
- Reanalysis from r_{main} triggers side-effects to g in new equation system

$\Longrightarrow$ New values for $g \in G$ without contributions from previous runs.

Usability

IDE integration via MagpieBridge [1], using server mode for Goblint:

- Communication IDE $\Longleftrightarrow$ Goblint via sockets
- Configuration is maintained
- Works without restart of analyzer and reparsing of unchanged code



Results

Thread-modular, partially context-sensitive analysis with intervals and race-detection performed on commits in *zstd*, *chrony*, *figlet* repositories.

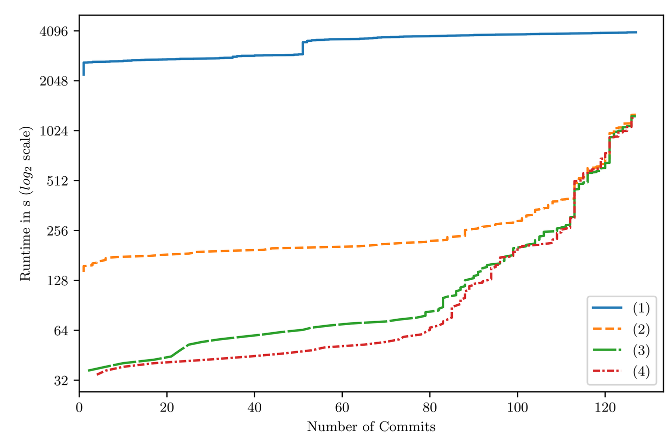


Figure 1: Cumulative distribution of commits analyzed within the given run time for setups (1)-(4) on *zstd*.

	(1)	(2)	(3)	(4)
from-scratch	✓	-	-	-
incremental	-	✓	✓	✓
incr. posts.	-	-	✓	✓
rel. destab	-	-	-	✓

Table 1: Features active in confs. (1)-(4).

	conf. (2)		conf. (3)		conf. (4)	
	solve	total	solve	total	solve	total
zstd	17.8	15.0	74.7	40.8	155.2	57.4
chrony	9.3	5.4	47.5	9.4	54.9	9.6
figlet	11.8	4.9	61.2	7.1	88.1	7.4

Table 2: Median speedups of *solving* (incl. postsolving) and *overall* run times achieved by configurations (2)-(4) compared to (1) on the benchmark repositories.

Conclusion

- Considerable speedups by interactive analysis
- Smaller overall speedups on smaller projects, due to other bottlenecks
- Restarting mitigates precision loss on flow-insensitive information

References

[1] L. Luo et al. "MagpieBridge: A General Approach to Integrating Static Analyses into IDEs and Editors (Tool Insights Paper)". In: *ECOOP 2019*. Ed. by A. F. Donaldson. Vol. 134. LIPIcs. Schloss Dagstuhl, 2019.

[2] H. Seidl et al. "Incremental Abstract Interpretation". In: *From Lambda Calculus to Cybersecurity Through Program Analysis*. Vol. 12065 LNCS. Springer, 2020, pp. 132–148.