Verification Witnesses

Marian Lingsch-Rosenfeld

Marian.Lingsch@sosy.ifi.lmu.de

Supervisor: Dirk Beyer

Collaborators: Marek Jankola (CONVEY) and Matthias Kettl

SoSy-Lab Software Systems

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

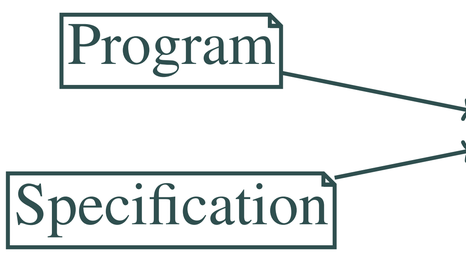
Verification witnesses [1, 2] accompany the results of a verifier and contain information about the verification process.

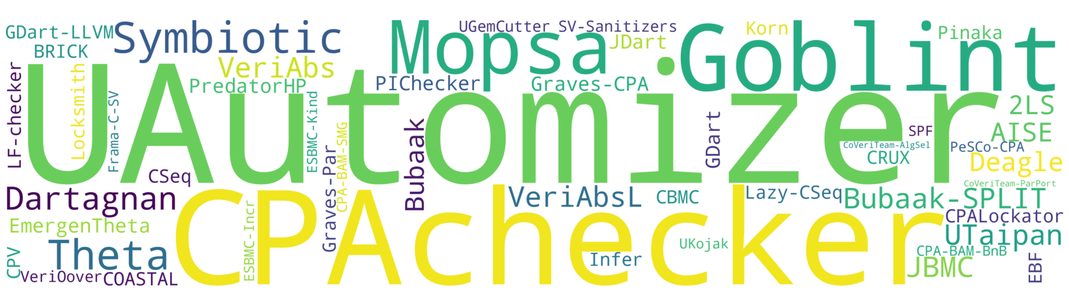
```
1 int main(void) {
2   int x = nondet();
3   int y = x ; int y = x + 1;
4
5   while (x < 1024) {
6     x++;
7     y++;
8   }
9
10  assert(x == y);
11 }
```

Example program

```
G ! assert fail
```

Specification





Verifiers participating in SV-COMP 2024; all export witnesses

Correctness witnesses encode invariants which aid in proving the program correct.

True
Correctness Witness

```
<...>
  content:
  - invariant:
      type: "loop_invariant"
      location:
        file_name: ""
        line: 5
        column: 3
        function: "main"
      value: "( y == x )"
      format: "c_expression"
```

Correctness witness version 2.0

False
Violation Witness

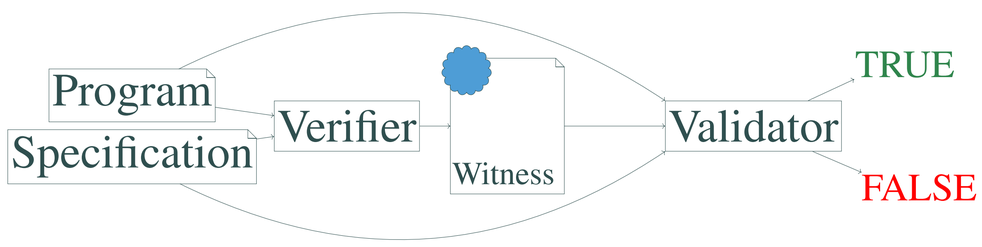
Violation witnesses represent a set of paths which lead to a violation of the specification.

```
<...>
  content:
  - segment:
    - waypoint:
        type: assumption
        location:
          line: 3
          file_name: ""
        constraint:
          value: "x == 1024"
  - segment:
    - waypoint:
        type: target
        location:
          line: 10
          file_name: ""
```

Violation witness version 2.0

Validation

Witness increase the confidence in the verification result by making its validation possible.



For example, LIV [5] creates a Hoare style proof from a witness. Which allows for independent validation of the verification result.

- $\{P\}s_0\{I\}$ (Reachability)

```
int x = nondet();
int y = x;
assert(x == y);
```

- $\{I \wedge C\}B\{I\}$ (Inductiveness)

```
int x = nondet();
int y = nondet();
assume(x == y &&
       x < 1024);
x++;
y++;
assert(x == y);
```

- $\{\neg C \wedge I\}\{Q\}$ (Safety)

```
int x = nondet();
int y = nondet();
assume(x == y);
assert(x == y);
```

Cooperation

Witnesses provide a common interface to exchange information between formal methods tools and theories.

Source	Communication Methods	Receiver	Restrictions
Automatic Verification	[3, 4, 7]	Automatic Verification	Understanding Witnesses
Automatic Verification	[6]	Deductive Verification	Programs with only Loops
Automatic Verification	[5]	Manual Proofs	Hoare style proofs; Programs with only Loops
Automatic Verification	Unknown	Interactive Theorem Proving	Unknown
Deductive Verification	Unknown	Interactive Theorem Proving	Unknown

References

[1] P. Ayaziová, D. Beyer, M. L. Rosenfeld, M. Spiessl, and J. Strejček. "Software Verification Witnesses 2.0". In: *Proc. SPIN*. Springer, 2024.

[2] D. Beyer, M. Dangl, D. Dietsch, M. Heizmann, T. Lemberger, and M. Tautschnig. "Verification Witnesses". In: *ACM Trans. Softw. Eng. Methodol.* 31.4 (2022), 57:1–57:69.

[3] D. Beyer, J. Haltermann, T. Lemberger, and H. Wehrheim. "Decomposing Software Verification into Off-the-Shelf Components: An Application to CEGAR". In: *Proc. ICSE*. ACM, 2022, pp. 536–548.

[4] D. Beyer, M. Lingsch-Rosenfeld, and M. Spiessl. "CEGAR-PT: A Tool for Abstraction by Program Transformation". In: *Proc. ASE*. IEEE, 2023, pp. 2078–2081.

[5] D. Beyer and M. Spiessl. "LIV: A Loop-Invariant Validation using Straight-Line Programs". In: *Proc. ASE*. IEEE, 2023, pp. 2074–2077.

[6] D. Beyer, M. Spiessl, and S. Umbricht. "Cooperation Between Automatic and Interactive Software Verifiers". In: *Proc. SEFM*. LNCS 13550. Springer, 2022, 111–128.

[7] D. Beyer and H. Wehrheim. "Verification Artifacts in Cooperative Verification: Survey and Unifying Component Framework". In: *Proc. ISoLA (1)*. LNCS 12476. Springer, 2020, pp. 143–167.