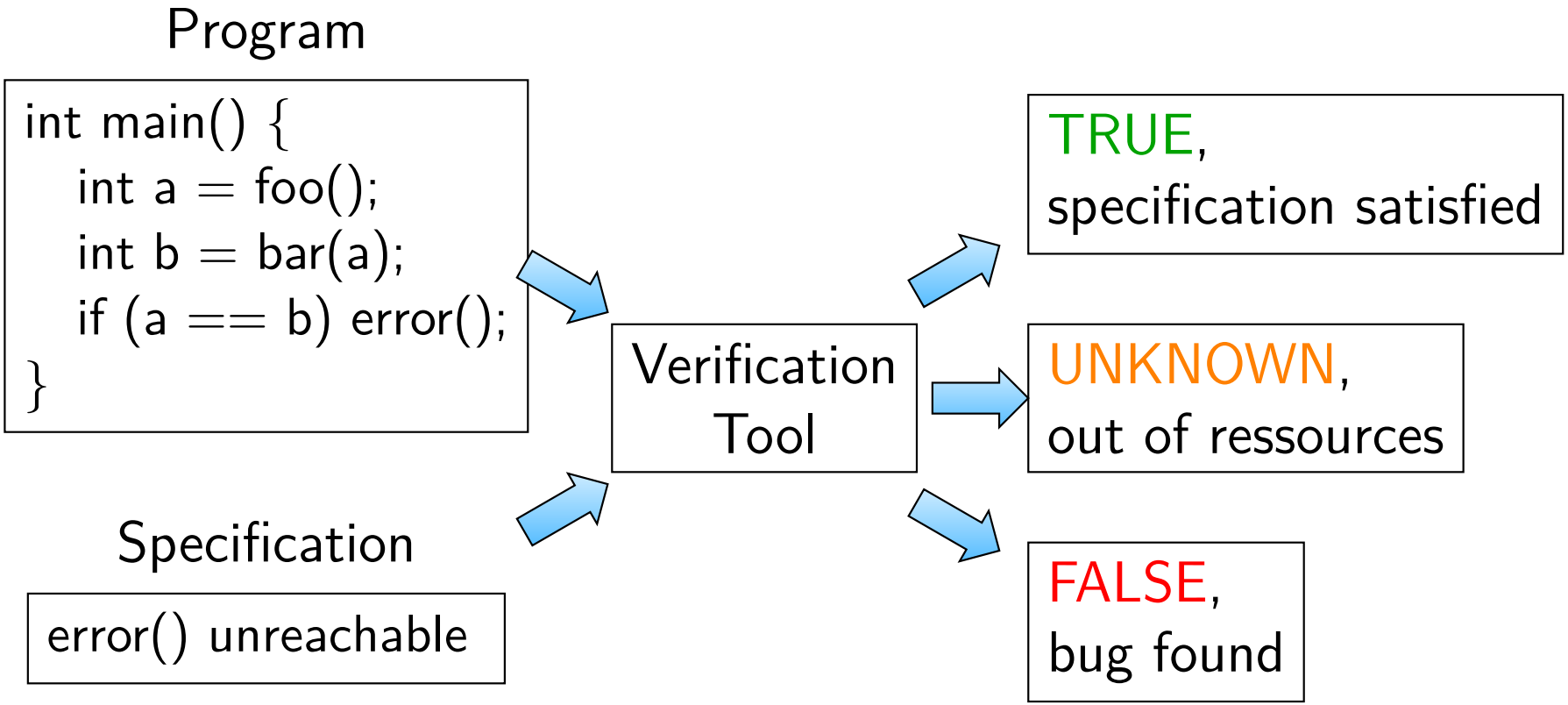
Applying CPAchecker to Large Explicit State Spaces

Dirk Beyer and Karlheinz Friedberger

LMU Munich, Germany

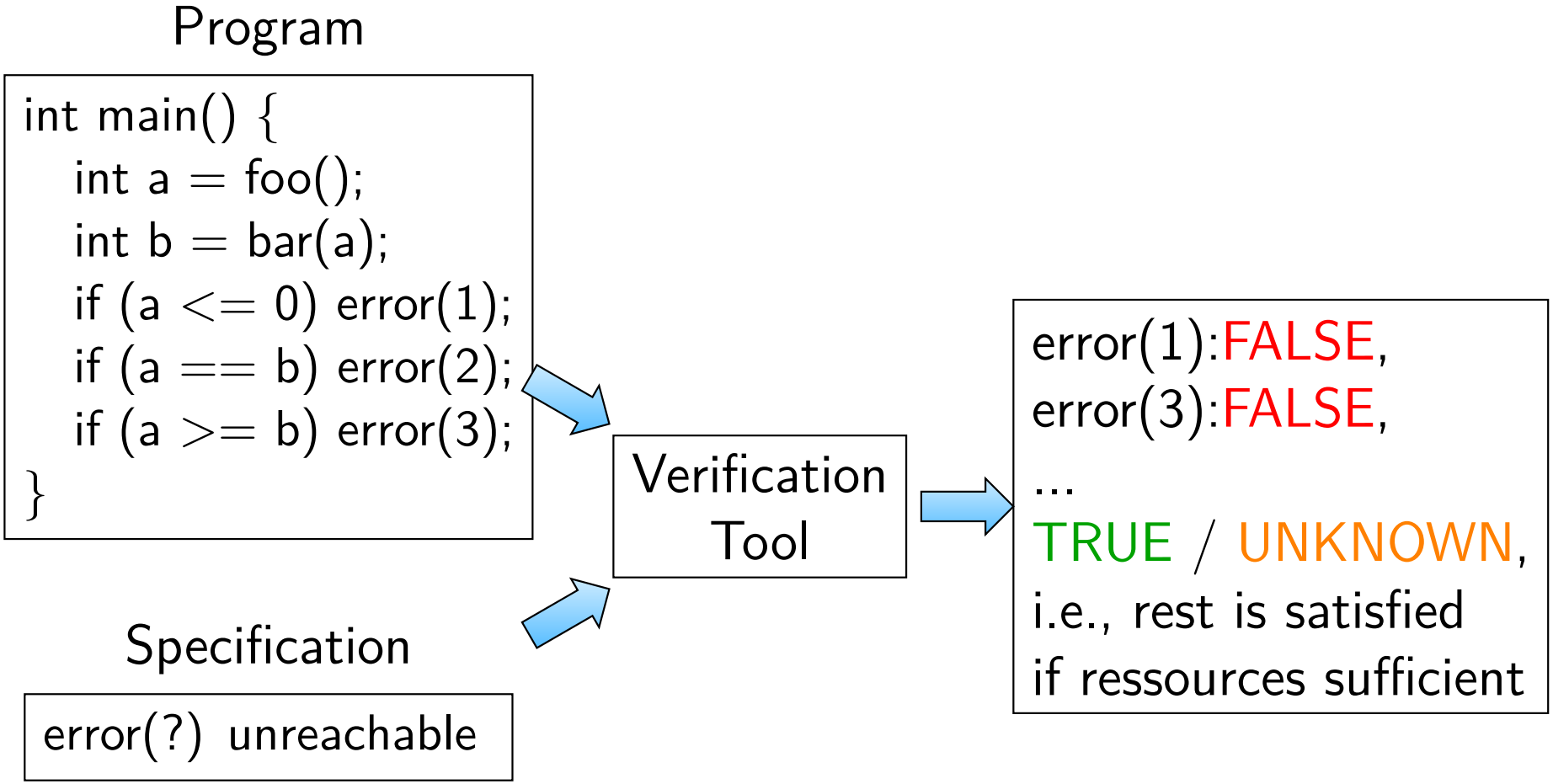
Nov 8, 2018, RERS, ISoLA

Single-Property Verification



Multi-Property Verification

Each RERS task contains 100 possible property violations
→ reduce repeated work by verifying all of them in one verification run



Configurations of CPAchecker for RERS'19

We tried the following approaches unsuccessfully:

- Precidate Analysis / BMC / kInduction [4]
  → large SMT formulas due to branching of control flow
  → non-linear arithmetic
- BDD analysis [1, 6]
  → non-linear arithmetic and pointers unsupported
- SMG analysis (for data-structures tasks) [7, 8]
  → very precise analysis for heap and stack memory
  → too expensive for large tasks
- BAM [9]
  → split task into blocks and analyze them separately
  → more overhead than benefit

Configurations of CPAchecker for RERS'19

Simple and most effective approach: Explicit-Value Analysis [5]

- Explicitly enumerate all reachable states
- Report all specification violations
- Simple specification automaton

Optimization:

- No CEGAR, full precision is needed anyway
- Large-block encoding to reduce memory consumption and runtime, i.e., only check for coverage at loopheads
- Combine with primitive pointer analysis

Specification for Optimized Explicit-Value Analysis

```
CONTROL AUTOMATON MultiErrors
INITIAL STATE Init;
STATE USEALL Init:
  MATCH {__VERIFIER_error($?)}
    -> PRINTONCE "$rawstatement in line $line"
       GOTO Init;
END AUTOMATON
```

- Based on Blast Query Language (SAS 2004, [2])
- Simply report all function calls of `__VERIFIER_error`
- Multi-property-verification:
  do not stop after first result, but explore whole state space

Environment and Results

- Intel i7-6700 CPU with 8 cores
- 25 GB RAM and 48 h CPU time for each of the nine tasks
- Small script converts CPAchecker output into RERS format

#	size	structure	# found violations	# proofs
10	small	plain	14	86
11		arithmetic	20	-
12		data structures	2	-
13	medium	plain	20	-
14		arithmetic	14	-
15		data structures	1	-
16	large	plain	37	-
17		arithmetic	26	-
18		data structures	2	-

Conclusion

- Very flexible specification language for automata
- No changes necessary to existing analyses and concepts

Future Work (until next year):

- Add SMT-based counterexample check
  to avoid false results from pointer analysis
- Add execution-based counterexample check [3]
- We need ClusterBAM!

References I

S. Apel, D. Beyer, K. Friedberger, F. Raimondi, and A. v. Rhein.
Domain types: Abstract-domain selection based on variable usage.
In *Proc. HVC*, LNCS 8244, pages 262–278. Springer, 2013.

D. Beyer, A. J. Chlipala, T. A. Henzinger, R. Jhala, and R. Majumdar.

The BLAST query language for software verification.
In *Proc. SAS*, LNCS 3148, pages 2–18. Springer, 2004.

D. Beyer, M. Dangl, T. Lemberger, and M. Tautschnig.
Tests from witnesses: Execution-based validation of verification results.

In *Proc. TAP*, LNCS 10889, pages 3–23. Springer, 2018.

D. Beyer, M. Dangl, and P. Wendler.
A unifying view on SMT-based software verification.
J. Autom. Reasoning, 60(3):299–335, 2018.

References II

D. Beyer and S. Löwe.
Explicit-state software model checking based on CEGAR and interpolation.
In *Proc. FASE*, LNCS 7793, pages 146–162. Springer, 2013.

D. Beyer and A. Stahlbauer.
Bdd-based software verification: Applications to event-condition-action systems.
STTT, 16(5):507–518, 2014.

K. Dudka, P. Peringer, and T. Vojnar.
Byte-precise verification of low-level list manipulation.
In *Proc. SAS*, LNCS 7935, pages 215–237. Springer, 2013.

P. Muller and T. Vojnar.
CPAlien: Shape analyzer for CPAchecker (competition contribution).
In *Proc. TACAS*, LNCS 8413, pages 395–397. Springer,

References III


D. Wonisch and H. Wehrheim.
Predicate analysis with block-abstraction memoization.
In *Proc. ICFEM*, LNCS 7635, pages 332–347. Springer, 2012.