# ULTIMATE AUTOMISET



☐ ultimate.informatik.uni-freiburg.de

☐ github.com/ultimate-pa/ultimate

Matthias Heizmann, Manuel Bentele, Daniel Dietsch, Xinyu Jiang, Dominik Klumpp, Frank Schüssele, Andreas Podelski

### **Features**

- Memory safety analysis
- Overflow detection
- Termination analysis using Büchi automata
- Nontermination analysis using geometric nontermination arguments
- LTL software model checking
- Bitprecise analysis
- IEEE 754 floating point analysis
- Error witnesses
- Correctness witnesses
- Error localization

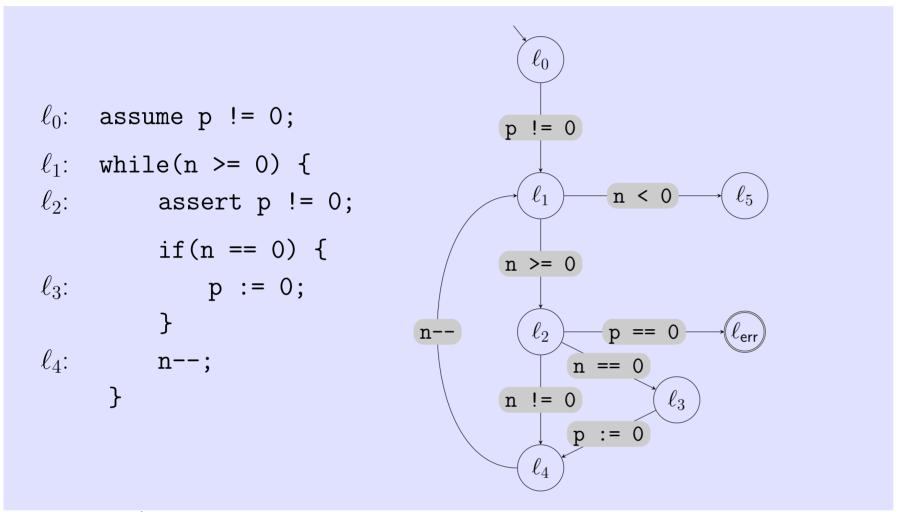
### **Techniques**

- On-demand trace-based decomposition
- Interprocedural analysis via nested word automata
- Theory-independent interpolation
- Refinement selection
- Configurable block encodings
- Multi SMT solver support
- Synthesis of ranking functions
- Efficient complementation of semi-deterministic Büchi automata
- (Nested word) automata minimization

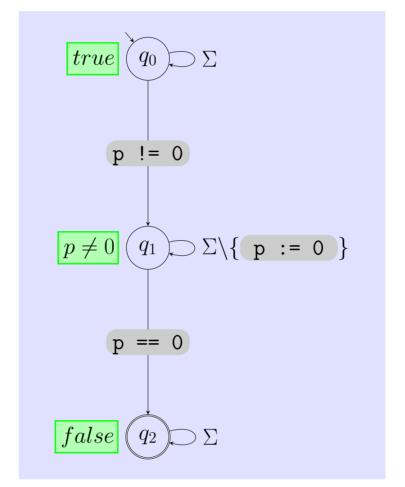
### ULTIMATE program analysis framework Controller Third-Party Libraries **Internal Libraries** Lib UltimateTest **CDTPlugin** org.eclipse.ui **SMTInterpol SMTSolverBridge** GuiGenerated PreferencePage Lib SmtLib UltimateCLI org.eclipse.jface Xercex UltimateGUI ACSLParser Lib JUnitUtil Lib Automata UltimateTest log4j CZT JavaCup Lib UltimateUtil ASTBuilder WebInterface Core ojAlgo srParse Source Analyzer Generators BoogieModSet Annotator AutomataScript TraceAbstraction SmtParser **IRSDependencies** LTL2Aut CodeCheck BoogiePrinter Concurrent Reaching Definitions TraceAbstraction WithAFAs CfgPrinter SpaceExParser Cookify BoogieParser Jung Visualization SyntaxChecker DSITransformer CDTParser ProcedureInline BlockEncoding TraceAbstractio WitnessPrinter Abstract **PEAtoBoogie** LassoRanker HeapSeparator RcfgBuilder AutomataScript Interpreter BuchiAutomizer

## Automata-theoretic proof of program correctness

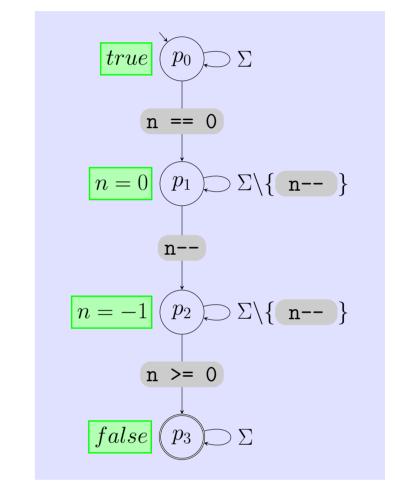
Program  $\mathcal{P}$  is correct because each error trace is infeasible, i.e. the inclusion  $\mathcal{P} \subseteq \mathcal{A}_1 \cup \mathcal{A}_2$  holds.



Program / automaton  $\mathcal{P}$  whose language is the set of error traces.

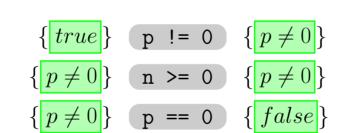


Automaton  $A_1$  whose language is a set of infeasible traces.



Automaton  $\mathcal{A}_2$  whose language is a set of infeasible traces.

- Alphabet: set of program statements
- $\Sigma = \{ p \neq 0, n < 0, n >= 0, p == 0, n == 0, n \neq 0, p == 0, n -- \}$
- $\bullet$  The language of  $\mathcal P$  is the set of error traces.
- In the first iteration, we analyze feasibility of the error trace  $\pi_1 = p = 0$  n >= 0 p == 0.  $\pi_1$  is infeasible. Via interpolation, we obtain the following Hoare triples.



We construct the automaton  $\mathcal{A}_1$  such that its language is the set of all traces whose infeasibility can be shown using the predicates true,  $p \neq 0$ , and false.

- Analogously, in the second iteration the automaton  $A_2$  is constructed.
- We check the inclusion  $\mathcal{P} \subseteq \mathcal{A}_1 \cup \mathcal{A}_2$  and conclude that each error trace is infeasible and hence  $\mathcal{P}$  is correct.

**Definition** Given an automaton  $\mathcal{A} = (Q, \delta, q_{\mathsf{init}}, Q_{\mathsf{final}})$  over the alphabet of program statements, we call a mapping that assigns to each state  $q \in Q$  a predicate  $\varphi_q$  a Floyd-Hoare annotation for automaton  $\mathcal{A}$  if the following implications hold.

$$(q, s, q') \in \delta \implies \{\varphi_q\} s \{\varphi_{q'}\}$$
 is a valid Hoare triple  $q = q_{\mathsf{init}} \implies \varphi_q = true$   $q \in Q_{\mathsf{final}} \implies \varphi_q = false$ 

**Theorem** If an automaton  $\mathcal{A}$  has a Floyd-Hoare annotation, then  $\mathcal{A}$  recognizes a set of infeasible traces.

### Interpolation with unsatisfiable cores

Level 1: "interpolation" via

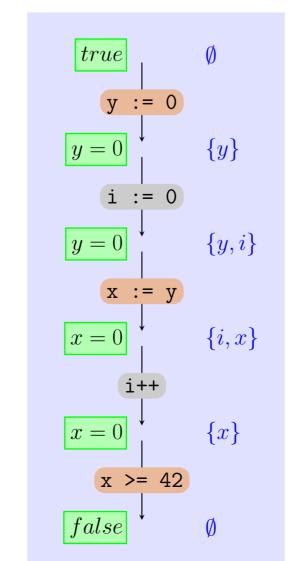
• strongest post

Level 2: interpolation via

- strongest post
- live variable analysis

Level 3: interpolation via

- strongest post
- live variable analysis
- unsatisfiable cores



Algorithm (for level 3)

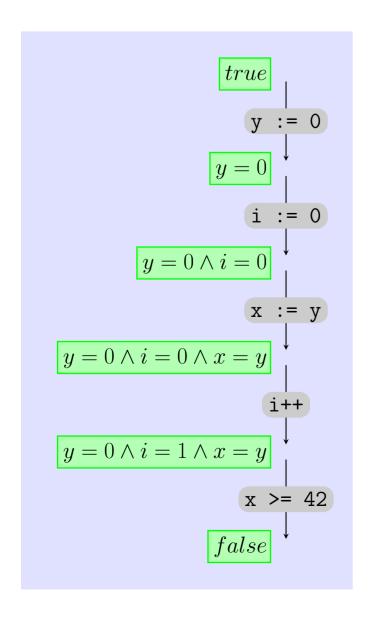
- Input: infeasible trace  $x_1, \ldots, x_n$  and unsatisfiable core  $UC \subseteq \{x_1, \ldots, x_n\}$ .
- Replace each statement that does not occur in UC by a skip statement or a havoc statement.

assume statement  $\psi \rightsquigarrow \text{skip}$  assignment statement  $\text{x:=t} \rightsquigarrow \text{havoc x}$ 

• Compute sequence of predicates  $\varphi_0, \ldots, \varphi_n$  iteratively using the strongest post predicate transformer sp.

$$\varphi_0 := true$$
  
$$\varphi_{i+1} := sp(\varphi_i, \pounds_{i+1})$$

- Eliminate each variable from predicate  $\varphi_i$  that is not live at position i of the trace.
- Output: sequence of predicates  $\varphi_0, \ldots, \varphi_n$  which is a sequence of interpolants for the infeasible trace  $\mathfrak{x}_1, \ldots, \mathfrak{x}_n$ .



 $true | \emptyset$  y := 0  $y = 0 \downarrow$  i := 0 x := y  $i = 0 \land x = 0 \downarrow$  x := y  $i + + \downarrow$   $x = 0 \downarrow$  x >= 42  $false \downarrow$   $\emptyset$