Ultimate Kojak

ultimate.informatik.uni-freiburg.de

github.com/ultimate-pa/ultimate

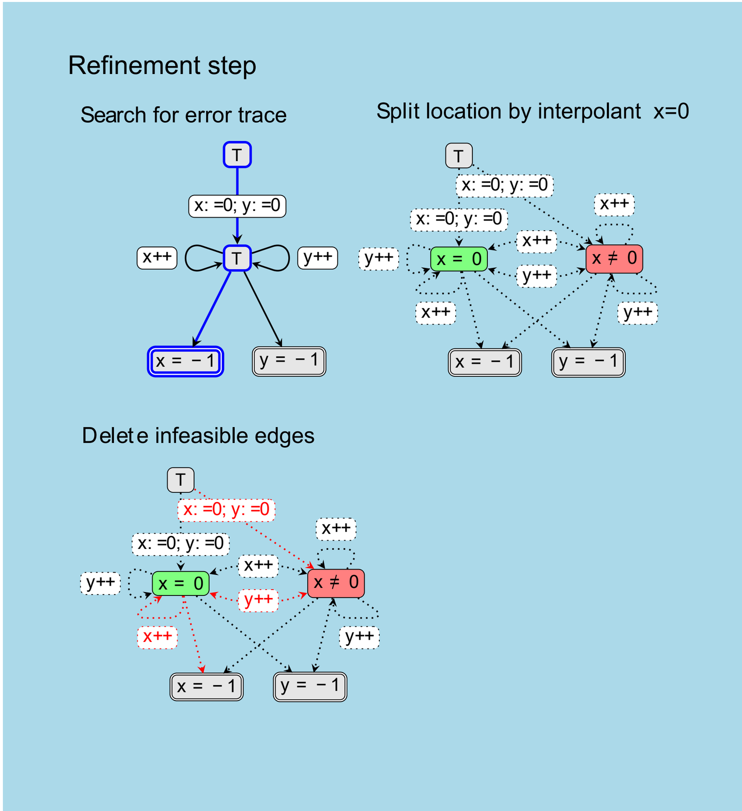
Daniel Dietsch, Marius Greitschus, Matthias Heizmann, Jochen Hoenicke, Alexander Nutz, Christian Schilling, Tanja Schindler

Features

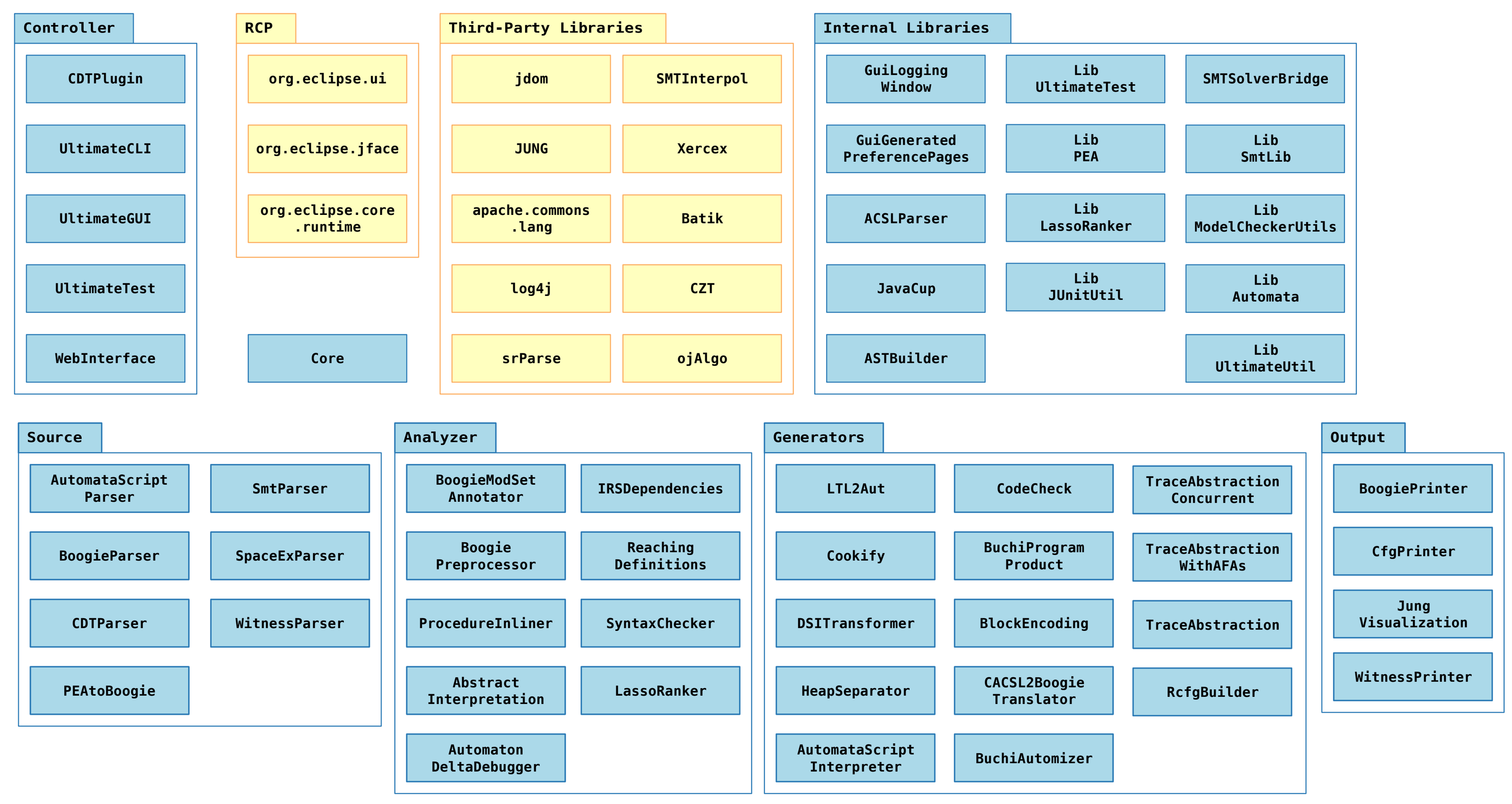
- Reachability analysis
- Memory safety analysis
- Bitprecise analysis
- IEEE 754 floating point analysis
- Error witnesses
- Correctness witnesses

Techniques

- Abstraction refinement
- Configurable block encodings
- Multi SMT solver support
- Newton-style interpolation



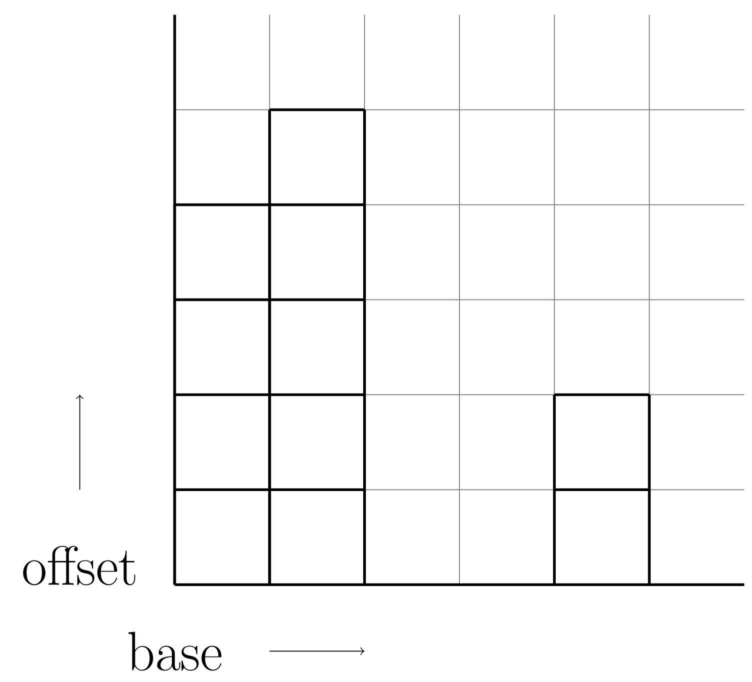
Ultimate program analysis framework



C memory model

Models dynamically allocated memory through Boogie arrays:

- `memory-[int|pointer|bitvector8|...]`: store memory contents
  - one array per used Boogie data type
  - two dimensional, a memory address has components "base" and "offset"
  - models disjointness of memory areas allocated by different malloc calls
- `valid`: store which base addresses are allocated
- `length`: store maximal offset at each base address
- "`*p` is a valid pointer dereference" $\iff$ `valid[p.base]` $\wedge$ `p.offset` $\leq$ `length[p.base]`
- "Program has no memory leaks" $\iff$ `valid` = `old(valid)` at the end of `main`



valid	T	T	F	F	T	F
length	4	5	-1	-1	2	-1

SMT solver integration

Hoare triple checks

"Is $\{P\}\,s\,\{Q\}$ a Hoare triple?"

Features:

- Simplify check if $(\mathit{variables}(P) \cup \mathit{variables}(\mathit{st})) \cap \mathit{variables}(Q) = \emptyset$.
  - often blocked because P, st and Q access the same array (but perhaps at different positions)
  - attempt to partition arrays via "alias analysis" (work in progress)
- Avoid checks with intricate predicates.
- Use incremental (push/pop) solver queries when possible, e.g., group checks that share the same precondition P.
- Abstract interpretation-based:
  Check if $\mathit{post}^{\#}(P^{\#}, \mathit{st}) \sqsubseteq Q^{\#}$ holds in some abstract domain.
- Unify equivalent predicates.
- Cache Hoare triples and implication between predicates.

Tree interpolation

- Interpolating solvers used by Ultimate: SMTInterpol, Z3
- Tree interpolation syntax example (procedures `foo`, `bar`):

```
(assert (! (..) :named foo-stm1))
(assert (! (..) :named foo-stm2))
(assert (! (..) :named bar-stm1))
(assert (! (..) :named bar-stm2))
(assert (! (..) :named foo-stm3))
(check-sat)
(get-interpolants (foo-stm1 foo-stm2 (bar-stm1 bar-stm2) foo-stm3))
```

Interface

- Java interface (currently only SMTInterpol)
- SMTLib2 interface
- Solvers in use at SV-COMP 2018: SMTInterpol, Z3, MathSat, CVC4
  as many as we can get!

Newton-style interpolation

- Input: infeasible trace $\mathit{st}_1, \ldots, \mathit{st}_n$, unsatisfiable core $\mathsf{UC} \subseteq \{\mathit{st}_1, \ldots, \mathit{st}_n\}$
- Replace statements not in UC:
  assume statement $\psi \rightsquigarrow$ `skip`
  assignment statement `x:=t` $\rightsquigarrow$ `havoc x`
- Compute sequence of predicates $\varphi_0, \ldots, \varphi_n$ iteratively using strongest post operator post

$$\varphi_0 := \mathit{true}$$
$$\varphi_{i+1} := \mathit{post}(\varphi_i, \mathit{st}_{i+1})$$

- Eliminate each variable from predicate φ_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 $\mathit{st}_1, \ldots, \mathit{st}_n$

	trace τ	state assertions for τ		interpolating trace $\tau^{\#}$	state assertions for $\tau^{\#}$
φ_0		true	φ_0		true
st_1	`b:=a`			`b:=a`	
φ_1		$a = b$	φ_1		$a = b$
st_2	`x:=0`			`havoc x`	
φ_2		$a = b \wedge x = 0$	φ_2		$a = b$
st_3	`havoc p`			`havoc p`	
φ_3		$a = b \wedge x = 0$	φ_3		$a = b$
st_4	`!a[p]`			`!a[p]`	
φ_4		$a = b \wedge x = 0 \wedge a[p] = \mathit{false}$	φ_4		$a = b \wedge a[p] = \mathit{false}$
st_5	`a[p]:=true`			`a[p]:=true`	
φ_5		$a = b[p := \mathit{true}] \wedge x = 0 \wedge a[p] = \mathit{true}$	φ_5		$a = b[p := \mathit{true}] \wedge a[p] = \mathit{true}$
st_6	`x:=x+1`			`havoc x`	
φ_6		$a = b[p := \mathit{true}] \wedge x = 1 \wedge a[p] = \mathit{true}$	φ_6		$a = b[p := \mathit{true}] \wedge a[p] = \mathit{true}$
st_7	`a[p]:=false`			`a[p]:=false`	
φ_7		$a = b \wedge x = 1 \wedge a[p] = \mathit{false}$	φ_7		$a = b \wedge a[p] = \mathit{false}$
st_8	`a!=b`			`a!=b`	
φ_8		false	φ_8		false