VL Semantik
Lehrstuhl für Software and Computational Systems
Dirk Beyer, Thomas Lemberger

Winter Term 2020/2021

Handout

2021/02/19

1 Ablauf

13:00 – 13:15	Intro
13:15 – 13:45	pySMT und BMC (Breakout Rooms)
13:45 – 13:55	Besprechung
13:55 – 14:00	Pause
14:00 – 14:10	Einführung CPAs
14:10 – 14:45	CPAs (Breakout Rooms)
14:35 – 14:45	Besprechung CPAs
14:45 – 14:50	Pause
14:50 – 15:20	CPAs (Breakout Rooms)
15:20 – 15:30	Nachbesprechung

1.1 Teambildung

2er-Teams. Wer kann Python?

2 SMT Solvers

Satisfiability modulo theories.

Theories:

- Arrays
- Arithmetic (Integer, Float, Bitvector)
- Undefined functions
- ...

3 Configurable Program Analysis

3.1 Semi-Lattice

Semi-lattice $\mathcal{E} = (E, \sqsubseteq, \sqcup, \top)$ over elements of a set E, if:

- $\sqsubseteq$: $E \times E$ partial order over E,
- every subset $M \subseteq E$ has a least upper bound $e \in E$,
- $\sqcup : E \times E \to E$ denotes the leat upper bound of two elements,
- top element $\top$ is the least upper bound of E.

3.2 CPAs

A CPA $\mathbb{D} = (D, \rightsquigarrow, \mathsf{merge}, \mathsf{stop})$ for a CFA (L, l_0, G) consists of the following components:

- Abstract domain $D = (C, \mathcal{E}, [\![\cdot]\!])$ with concrete states C, semi-lattice $\mathcal{E} = (E, \sqsubseteq, \sqcup, \top)$, and concretization function $[\![\cdot]\!] : E \to 2^C$
- Abstract transfer relation $\rightsquigarrow$: $E \times G \times E$ assigns to each abstract state $e \in E$ possible abstract successors $e' \in E$, labelled with a corresponding CFA edge $g \in G$.
- Merge operator $\mathsf{merge} : E \times E \to E$ combines two abstract states into a new one
- Termination check $\mathsf{stop} : E \times 2^E \to \mathbb{B}$ checks whether an abstract state is already covered by a set of given abstract states

Some CPAs you should know:

1. Location CPA
2. Observer Analysis
3. Value Abstraction
4. Predicate Abstraction

3.3 CPA Algorithm

Algorithm 2 $CPA(\mathbb{D}, e_0)$

Input: a CPA $\mathbb{D} = (D, \rightsquigarrow, \mathsf{merge}, \mathsf{stop})$,
an initial abstract state $e_0 \in E$, where E denotes the set of elements of the lattice of D
Output: a set of reachable abstract states
Variables: a set $\mathsf{reached} \subseteq E$, a set $\mathsf{waitlist} \subseteq E$

waitlist $:= \{e_0\}$
reached $:= \{e_0\}$
while waitlist $\neq \{\}$ **do**
  choose e from waitlist
  waitlist $:=$ waitlist $\setminus \{e\}$
  for each e' with $e \rightsquigarrow e'$ **do**
    for each $e'' \in$ reached **do**
      // combine with existing abstract state
      $e_{new} :=$ merge(e', e'')
      if $e_{new} \neq e''$ **then**
        waitlist $:=$ (waitlist $\cup \{e_{new}\}) \setminus \{e''\}$
        reached $:=$ (reached $\cup \{e_{new}\}) \setminus \{e''\}$
    if $\neg$ stop$(e'$, reached$)$ **then**
      waitlist $:=$ waitlist $\cup \{e'\}$
      reached $:=$ reached $\cup \{e'\}$
return reached

3.4 Linear Temporal Logic (LTL)

As a reminder, the syntax of LTL:

Formula ϕ ::= true | false | A	atomic propositions
$\mid \neg\phi \mid \phi \wedge \psi \mid \ldots$	junctors over LTL formulae
$\mid \circ\ \phi \quad (\mathcal{N}\ \phi)$	ϕ is true in the next state (/next time step)
$\mid \diamond\ \phi \quad (\mathcal{F}\ \phi)$	ϕ is true sometime between now and the infinite future
$\mid \square\ \phi \quad (\mathcal{G}\ \phi)$	ϕ is true all the time, from now on
$\mid \phi \mathcal{U} \psi$	ϕ is true until ψ is true, and ψmust be true at some point
$\mid \phi \mathcal{W} \psi$	ϕ is true until ψ is true, and ψmay always stay false

An LTL formula is evaluated over an infinite sequence of steps. In each step, each atomic proposition may change its value.

The definition of a (time) step is arbitrary. In our application, a time step is often defined as one transition in the CFA.

4 Verification-Result Witnesses

https://github.com/sosy-lab/sv-witnesses

4.1 Violation Witnesses

4.2 Correctness Witnesses