Handout

2025/02/28

1 Configurable Program Analysis

1.1 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.

1.2 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

1.3 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 := {e0}
reached := {e0}
while waitlist ≠ {} do
   choose e from waitlist
   waitlist := waitlist \ {e}
   for each e' with e⇝e' do
      for each e'' ∈ reached do
         // combine with existing abstract state
         e_new := merge(e', e'')
         if e_new ≠ e'' then
            waitlist := (waitlist ∪ {e_new}) \ {e''}
            reached := (reached ∪ {e_new}) \ {e''}
      if ¬ stop(e', reached) then
         waitlist := waitlist ∪ {e'}
         reached := reached ∪ {e'}
return reached
```