CS 5/6110, Software Correctness Analysis, Spring 2022

Ganesh Gopalakrishnan
School of Computing
University of Utah
Salt Lake City, UT 84112

Summary of the basics of Static Analysis
(0) where we left off was Live Variable Analysis
Discuss that using slide-deck #4 of Andersson-Moller
(1) Basics from PhD thesis by Dorra Ben Khalifa
https://tel.archives-ouvertes.fr/tel-03509266
(2) from Moller/Schwartzbach
(3) from the SPARTA team (Android)

Purpose of this lecture

- To familiarize you with resources to learn and practice static analysis (these are more practical tools below)
  - Eva tool (part of Frama-C)
  - ASTREE tool (academic groups have used it)
    - industrial-strength
      - Used by Airbus, French companies, …
  - Coverity, Galois Inc tools, …
    - Similar deal in the US
  - SPARTA
    - Facebook's academic incubator
    - Has a good set of tutorials

Purpose of this lecture

- To familiarize you with resources to learn and pedagogical static analysis
  - Schwartzbach's notes
    - SCALA implementation
    - LLVM Implementation
  - https://cs.au.dk/~amoeller/spa/

Excerpts from the Ben Khalifa thesis

2.2 Static Analysis by Abstract Interpretation

Static program analysis aims at automatically determining whether a program satisfies some particular properties such as "the program never dereferences a null pointer", "the program never divides by zero", "the user-specified assertions are never violated", etc. Abstract interpretation [CC77a, CH78, CC92] provides the mathematical theory to design such analysis. It consists of a general theory for approximating the behavior of programs, developed by Patrick Cousot and Radhia Cousot in the late 1970s, as a unifying framework for static program analysis. Abstract interpretation gathers the concepts necessary to build an approximate static analysis.

Excerpts from the Ben Khalifa thesis

Definition 2.5 (Relation). A binary relation $\mathcal{R}$ between two sets $\mathcal{A}$ and $\mathcal{B}$ is a subset of the Cartesian product $\mathcal{A} \times \mathcal{B}$. We often write $x \mathcal{R} y$ for $(x, y) \in \mathcal{R}$. We present hereafter some important properties which may hold for a binary relation $\mathcal{R}$ over a set $\mathcal{S}$:

- $\forall x \in \mathcal{S} : x \mathcal{R} x$ (reflexivity),
- $\forall x \in \mathcal{S} : \neg(x \mathcal{R} x)$ (irreflexivity),
- $\forall x, y \in \mathcal{S} : x \mathcal{R} y \Rightarrow y \mathcal{R} x$ (symmetry),
- $\forall x, y \in \mathcal{S} : x \mathcal{R} y \wedge y \mathcal{R} x \Rightarrow x = y$ (anti-symmetry),
- $\forall x, y, z \in \mathcal{S} : x \mathcal{R} y \wedge y \mathcal{R} z \Rightarrow x \mathcal{R} z$ (transitivity),
- $\forall x, y \in \mathcal{S} : x \mathcal{R} y \vee y \mathcal{R} x$ (totality).

Ben Khalifa...

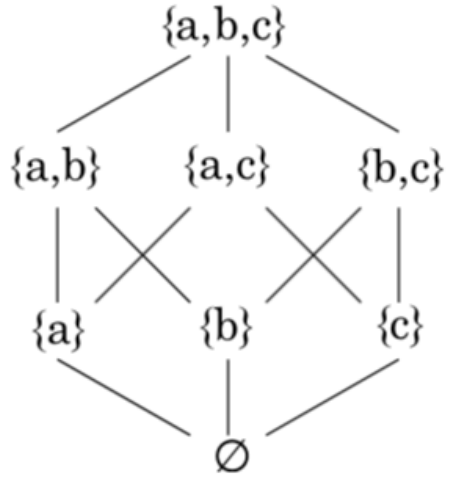

Figure 2.3: Hasse diagram for the partial order set of $(\mathcal{P}(a,b,c),\subseteq)$.

Definition 2.6 (Partial Order, Poset). A Partial order $\sqsubseteq$ on a set $\mathcal{P}$ is a relation $\sqsubseteq \in \mathcal{P} \times \mathcal{P}$ that is reflexive, anti-symmetric and transitive. A partial order set, or poset, $(\mathcal{P},\sqsubseteq)$ is a set $\mathcal{P}$ equipped by a partial order $\sqsubseteq$.

Definition 2.7 (Lower and Upper Bounds). Let $(\mathcal{P},\sqsubseteq)$ be a poset, and $\mathcal{S} \subseteq \mathcal{P}$. An element $u \in \mathcal{P}$ is an *upper bound* of $\mathcal{S}$ if $\forall s \in \mathcal{S}, s \sqsubseteq u$. The element u is the *least upper bound*, or *join*, of $\mathcal{S}$, denoted by $\sqcup\mathcal{S}$, if $u \sqsubseteq u'$ for each upper bound u' of $\mathcal{S}$. Similarly, the element $l \in \mathcal{P}$ is a *lower bound* of $\mathcal{S}$ if $\forall s \in \mathcal{S}, u \sqsubseteq s$. The element l is the *greatest lower bound*, or *meet*, of $\mathcal{S}$, denoted by $\sqcap\mathcal{S}$, if $l' \sqsubseteq l$ for each lower bound l' of $\mathcal{S}$.

Ben Khalifa…

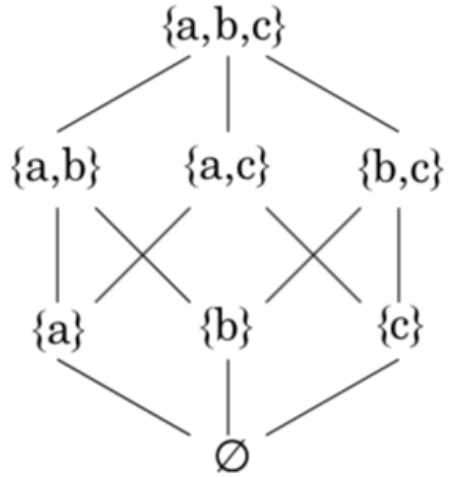

Figure 2.3: Hasse diagram for the partial order set of $(\mathcal{P}(a,b,c),\subseteq)$.

Definition 2.8 (Chain and Complete Partial Order). Let $(\mathcal{P},\sqsubseteq)$ be a poset. A *chain* $\mathcal{C} = (x_i)_{i\in\mathbb{N}}$ is a monotone sequence of elements of $\mathcal{P}$: $x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq x_{n-1} \sqsubseteq \cdots$. A *complete partial order (CPO)* is a poset $(\mathcal{P},\sqsubseteq)$ such that $\mathcal{P}$ has a least element $\bot$ and every chain $\mathcal{C}$ has a least upper bound $\sqcup\mathcal{C}$.

Definition 2.9 (Lattice). A lattice $(\mathcal{P}, \sqsubseteq, \sqcup, \sqcap)$ is a poset where each pair of elements $a, b \in \mathcal{P}$, has a least upper bound denoted by $a \sqcup b$, and a greatest lower bound denoted by $a \sqcap b$. If it exists, the least element $\sqcup\mathcal{P}$ is denoted $\bot$, called *bottom*. The greatest element $\sqcap\mathcal{P}$ is denoted $\top$, called *top*.

Ben Khalifa...

Example 2.4. An integer interval lattice can be constructed as follows:

$$(\{[a,b] | a,b \in \mathbb{Z}, a \leqslant b\} \cup \{\bot\}, \subseteq, \sqcup, \cap) \ .$$

Here, the set of integers $[a,b]$ is comprised between a and b with $a \leqslant b$. The smallest element $\bot$ represents the empty set $\varnothing$. The partial order is $\subseteq$ and $\cap$ is the greatest lower bound, as intervals are closed under intersection. However, they are not closed under set union, hence, the least upper bound can be defined as $[a,b] \sqcup [a',b'] = [\min(a,a'), \max(b,b')]$, while $\exists x : x \sqcup \bot = \bot \sqcup x = x$. Indeed, $[a,b] \sqcup [a',b']$ computes the smallest interval containing intervals $[a,b]$ and $[a',b']$.

Ben Khalifa...

Definition 2.10 (Complete Lattice). A complete lattice $(\mathcal{P}, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ is a poset such that for any subset of $\mathcal{P}$, possibly infinite, the least upper bound and the greatest lower bound are defined. Noting that in a complete lattice $\top$ and $\bot$ always exist.

Example 2.5. The integer interval lattice of Example 2.4 is not a complete lattice as the infinite family of intervals $\{[0, i] \mid i \geqslant 0\}$ has no least upper bound.

Ben Khalifa...

Definition 2.11 (Monotonic Function). Let $(\mathcal{P}_1, \sqsubseteq_1)$ and $(\mathcal{P}_2, \sqsubseteq_2)$ be two posets. A function $f : \mathcal{P}_1 \to \mathcal{P}_2$ is said to be *monotonic* if and only if:

$$\forall x, y \in \mathcal{P}_1, x \sqsubseteq_1 y \Rightarrow f(x) \sqsubseteq_2 f(y) \ . \tag{2.9}$$

Definition 2.12 (Continuous Function). Let $(\mathcal{P}_1, \sqsubseteq_1)$ and $(\mathcal{P}_2, \sqsubseteq_2)$ be two posets. A function $f : \mathcal{P}_1 \to \mathcal{P}_2$ is said to be *continuous* if it preserves existing least upper bounds of chains that is, for each chain $\mathcal{C} \subseteq \mathcal{P}_1$, if $\sqcup\mathcal{C}$ exists then we have:

$$f(\sqcup\mathcal{C}) = \sqcup\{f(x) | x \in \mathcal{C}\} \ . \tag{2.10}$$

Ben Khalifa...

2.2.2 Fixpoints

Once a language includes loops, the analysis needs to build a loop invariant, holding before entering the loop and after each iteration, upon re-entering the body. This invariant corresponds to a *fixpoint*. In the following, we recall the fundamental theorems due to Alfred Tarski [Tar55] and Stephen Cole Kleene [Bir67].

Definition 2.13 (Fixpoint Computation). Let $(\mathcal{P}, \sqsubseteq, \sqcup, \sqcap)$ be a lattice and let $F : \mathcal{P} \to \mathcal{P}$. An element $x \in \mathcal{P}$ is called a *fixpoint* of F if $F(x) = x$. Similarly, it is called a *pre-fixpoint* if $x \sqsubseteq F(x)$, and a *post-fixpoint* if $F(x) \sqsubseteq x$. If there exists, the *least fixpoint* of F, denoted by $\mathsf{lfp}(F)$, is a fixpoint of F such that, for every fixpoint $x \in \mathcal{P}$ of F, $\mathsf{lfp}(F) \sqsubseteq x$. The *greatest fixpoint* of F denoted by $\mathsf{gfp}(F)$ is defined similarly.

Ben Khalifa...

Theorem 2.1 (Tarski's Fixpoint Theorem). The set of fixpoints of a monotonic function $F : \mathcal{P} \to \mathcal{P}$ over a complete lattice is also a complete lattice.

Proof. By Tarski in [Tar55]. ■

In particular, Tarski's theorem implies that a monotonic function among a complete lattice has a least fixpoint lfp.

Theorem 2.2 (Kleene's Fixpoint Theorem). Let $(\mathcal{P}, \sqsubseteq)$ be a CPO and let $F : \mathcal{P} \to \mathcal{P}$ be a continuous function. Then F has a least fixpoint such that

$$\mathsf{lfp}(F) = \sqcup\{F^i(\bot) \mid i \in \mathbb{N}\} \ . \tag{2.11}$$

Proof. Found in [Cou78]. ■

Ben Khalifa…

```
void main(){
    int x = 0;          // 1
    while(x < 100) {    // 2
      x = x + 1;        // 3
    }                   // 4
}
```

$$x_1 = [0,0]$$
$$x_2 =]-\infty, 99] \cap (x_1 \cup x_3)$$
$$x_3 = x_2 + [1,1]$$
$$x_4 = [100, +\infty[\cap (x_1 \cup x_3)$$

Figure 2.4: A simple integer loop and its semantic equations.

Ben Khalifa…

iteration 1:

$$
\begin{aligned}
x_2^1 &=]-\infty, 99] \cap ([0,0] \cup \bot) \\
&= [0,0] \\
x_3^1 &= [0,0] + [1,1] \\
&= [1,1] \\
x_4^1 &= [100, +\infty[\cap ([0,0] \cup [1,1]) \\
&= \bot
\end{aligned}
$$

iteration 2:

$$
\begin{aligned}
x_2^2 &=]-\infty, 99] \cap ([0,0] \cup [1,1]) \\
&= [0,1] \\
x_3^2 &= [0,1] + [1,1] \\
&= [1,2] \\
x_4^2 &= [100, +\infty[\cap ([0,0] \cup [1,2]) \\
&= \bot
\end{aligned}
$$

iteration $i+1 (i < 100)$:

$$
\begin{aligned}
x_2^{i+1} &=]-\infty, 99] \cap ([0,0] \cup [1,i]) \\
&= [0,i] \\
x_3^{i+1} &= [0,i] + [1,1] \\
&= [1,i+1] \\
x_4^{i+1} &= [100, +\infty[\cap ([0,0] \cup [1,i]) \\
&= \bot
\end{aligned}
$$

Ben Khalifa...

Definition 2.16 (Widening). Let $(\mathcal{D}, \sqsubseteq)$ be a poset. A widening operator $\triangledown : (\mathcal{D} \times \mathcal{D}) \to \mathcal{D}$ is such that:

1. $\forall x, y \in \mathcal{D}$, we have $x \sqsubseteq x \triangledown y$ and $y \sqsubseteq x \triangledown y$;
2. for all increasing chains $x_0 \sqsubseteq x_1 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq \ldots$, the increasing chain

$$y_0 = x_0$$

$$\forall n \in \mathbb{N}, y_{n+1} = y_n \triangledown x_{n+1}$$

is ultimately stationary, $\exists l \geqslant 0 : \forall j \geqslant l : y_j = y_l$.

■

Ben K

Example 2.9. Taking again Example 2.7, the widening operator $\triangledown$ is defined by:

$$[a,b] \triangledown [a',b'] = [c,d] \quad with \quad c = \begin{cases} -\infty & \text{if } a' \leqslant a \\ a & \text{otherwise} \end{cases} \; et \; d = \begin{cases} +\infty & \text{if } b \leqslant b' \\ b & \text{otherwise} \end{cases}.$$

Narrowing helps to recover precision lost by widening steps. It is used to enforce or accelerate the convergence of decreasing iteration sequences. It is defined as follows:

Definition 2.17 (Narrowing). Let $(\mathcal{D},\sqsubseteq)$ be a poset. A narrowing operator $\triangle : (\mathcal{D} \times \mathcal{D}) \to \mathcal{D}$ is such that:

1. for all element $x, y \in \mathcal{D}$, if $x \sqsupseteq y$ we have $x \sqsupseteq (x \triangle y) \sqsupseteq y$;
2. for all decreasing chains $x_0 \sqsupseteq x_1 \sqsupseteq \ldots \sqsupseteq x_n \sqsupseteq \ldots$, the decreasing chain

$$y_0 = x_0$$

$$\forall n \in \mathbf{N}, y_{n+1} = y_n \triangle x_{n+1}$$

is ultimately stationary, $\exists l \geqslant 0 : \forall j \geqslant l : y_j = y_l$.

■

Example 2.10. The narrowing operator $\triangle$ on the interval domain of Example 2.7 is:

$$[a,b] \triangle [a',b'] = [c,d] \quad with \quad c = \begin{cases} a' & \text{if } a = -\infty \\ a & \text{otherwise} \end{cases} \; et \; d = \begin{cases} b' & \text{if } b = +\infty \\ b & \text{otherwise} \end{cases}.$$

Ben Khalifa…

Widening steps:

$$
\begin{aligned}
x_2^{10} &= [0,9] \triangledown [0,10] \\
&= [0,\infty[\\
x_3^{10} &= [0,\infty[+ [1,1] \\
&= [1,\infty[\\
x_4^{10} &= [100,\infty[\cap ([0,0] \cup [1,\infty[) \\
&= [100,\infty[
\end{aligned}
$$

Narrowing steps:

$$
\begin{aligned}
x_2^{11} &= [0,\infty[\triangle [0,99] \\
&= [0,99] \\
x_3^{11} &= [0,99] + [1,1] \\
&= [1,100] \\
x_4^{11} &= [100,\infty[\cap ([0,0] \cup [1,100]) \\
&= [100,100]
\end{aligned}
$$

Figure 2.6: Widening and narrowing steps of the example of Figure 2.4

A good intro to widening and narrowing in the context of SPARTA (of Facebook) appears in https://youtu.be/_fA7vkVJhF8

Widening / Narrowing

- Widening in Sparta seems to be based on existing program-level constraints
  - Apparently not practical always
  - See Slide 23 of Moller/Andersson
    - McCarthy's "91 function" !
    - http://www.cs.us.es/~mjoseh/pub/Proving_termination_with_multiset_orderings_in_PVS.pdf
    - https://www.cs.umd.edu/~mvz/handouts/wift-tutorial.pdf
- Narrowing "not implemented" in Sparta

Cool things in the SPARTA talk

Making things fast: Bourdoncle's Algorithm

Classic dataflow analysis: Every time the variables in any block change, repeat the analysis on **all** basic blocks in method until fixed point is reached

To get to a fixed point more quickly, we want to iterate on innermost loops first

To do that, we need to decompose the CFG into nested loops:

"Tarjan's SCC algorithm applied recursively"

Parallel implementation available soon!