Special topics in Logic and Security

Master Year II, Sem. I, 2022-2023

Ioana Leuștean
FMI, UB

Table of contents

References for static analysis

1 Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, USA, January 1979, pages 269–282. ACM, 1979.

2 A. Moller, M. I. Schwartzbach, Static Program Analysis, `https://cs.au.dk/~amoeller/spa/`

"Static program analysis is the art of reasoning about the behavior of computer programs without actually running them."

Static Analysis: motivating example

Live variable analysis

Live variables at a program point are those for which there is a subsequent use of the current value before a redefinition.

Consider the assignment

```
(I) z = x + y
```

The variables that are *live in I* (just before the instruction I) are:

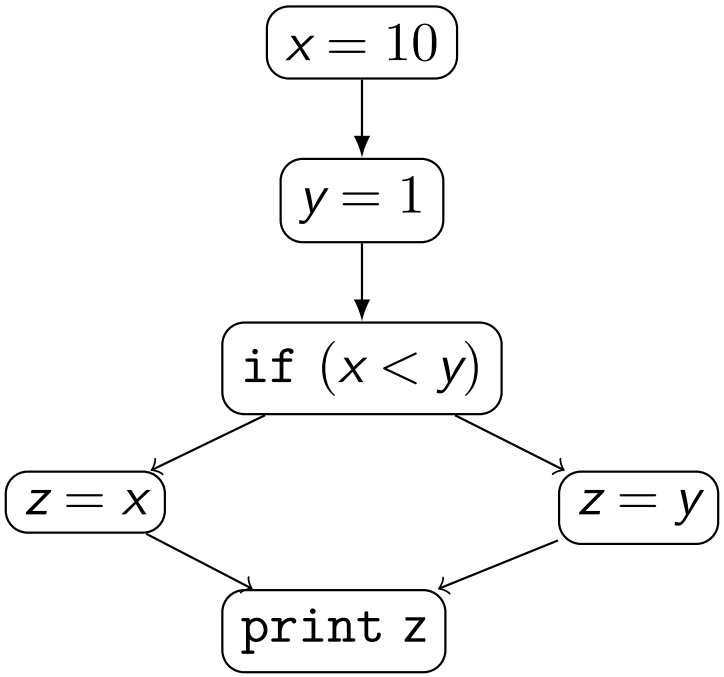
- the variables x and y because they are read,
- the variables that are live in the subsequent instructions, with the exception of z (since z is redefined).

Live variable analysis is an example of *static analysis*, which aims to analyze the properties of a program *without running it*.

Live variable analysis

Live variables at a program point P are those for which there is a subsequent use of he current value before a redefinition.

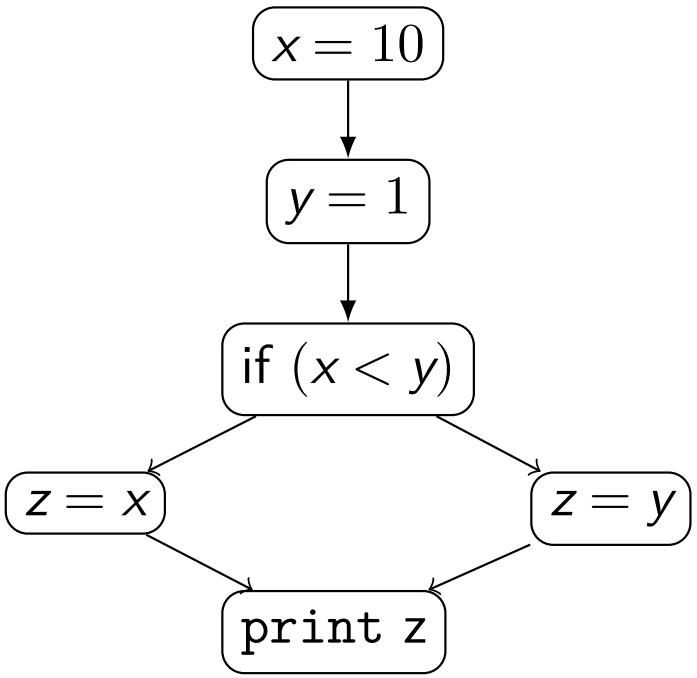
```
x = 10;
y = 1;
if  (x < y) z = x;
     else   z = y;
print z
```



Control Flow Graphs

Live variable analysis is *flow-sensitive*, meaning the order of statements affects the results. In order to solve the problem, our constraints are determined using the *Control Flow Graph* (CFG).

```
x = 10;
y = 1;
if  (x < y) z = x;
     else   z=y;
print z
```



The CFG provides information about the flow of control between the basic blocks of a program.

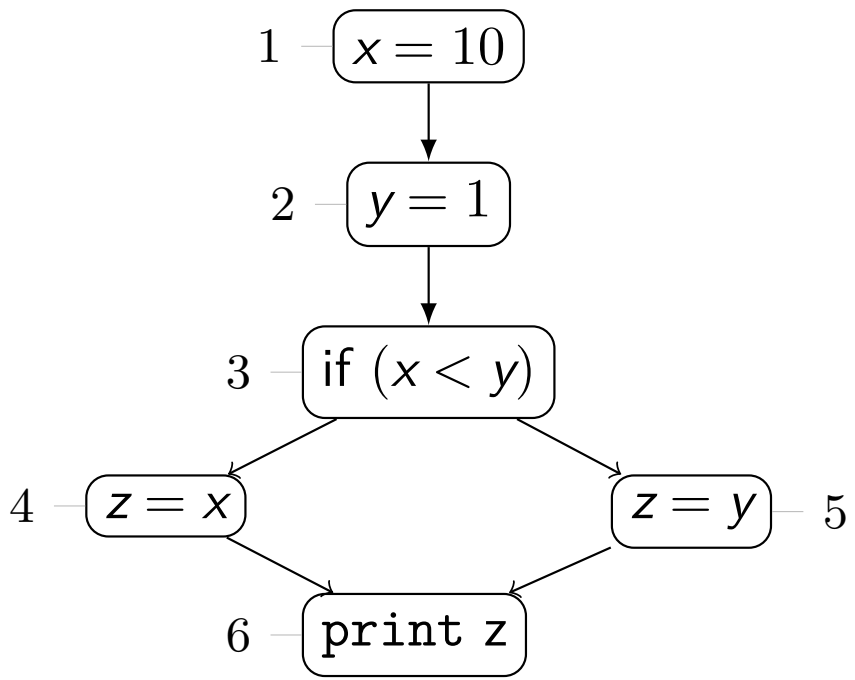
Live variable analysis

Live variables at a program point are those for which there is a subsequent use of the current value before a redefinition.

For a node n in CFG we denote by v_n the set of variables that are *live in* n (just before the instruction from n)

$$v_n = \left(\bigcup\{v_i \mid i \text{ succesor of } n\} \setminus \mathit{Written}(n)\right) \cup \mathit{Read}(n)$$

- $\mathit{Written}(n)$ is the set of variables that are redefined in n,
- $\mathit{Read}(n)$ is the set of variables that are read in n.

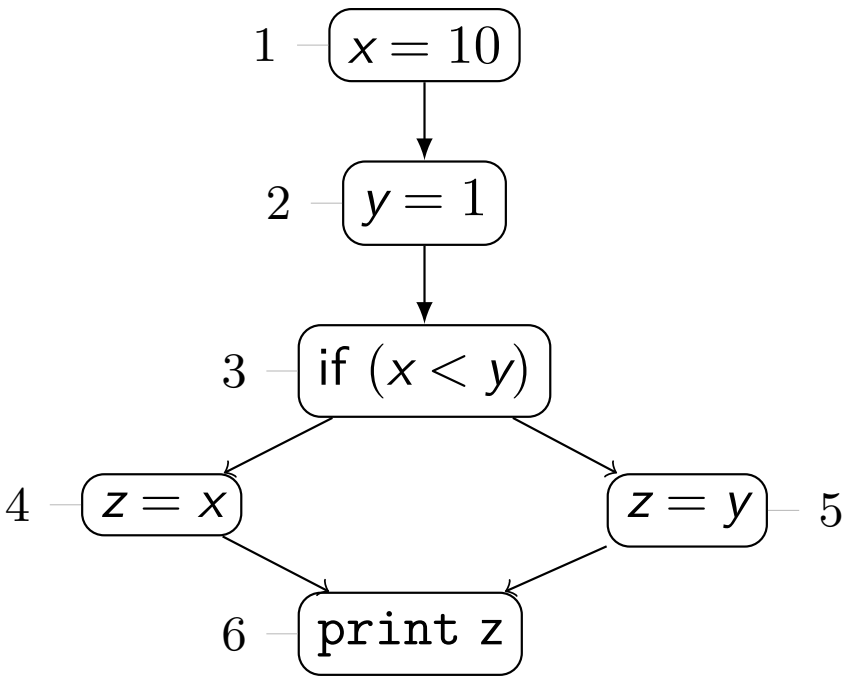


Live variable analysis

$$v_n = \left(\bigcup \{v_i \mid i \text{ succesor of } n\} \setminus \mathit{Written}(n)\right) \cup \mathit{Read}(n)$$

Constraints:

$$\begin{array}{lcl} v_1 & = & v_2 \setminus \{x\} \\ v_2 & = & v_3 \setminus \{y\} \\ v_3 & = & v_4 \cup v_5 \cup \{x, y\} \\ v_4 & = & (v_6 \setminus \{z\}) \cup \{x\} \\ v_5 & = & (v_6 \setminus \{z\}) \cup \{y\} \\ v_6 & = & \{z\} \end{array}$$



Live variable analysis

Constraints:

$$
\begin{array}{lcl}
v_1 & = & v_2 \setminus \{x\} \\
v_2 & = & v_3 \setminus \{y\} \\
v_3 & = & v_4 \cup v_5 \cup \{x, y\} \\
v_4 & = & (v_6 \setminus \{z\}) \cup \{x\} \\
v_5 & = & (v_6 \setminus \{z\}) \cup \{y\} \\
v_6 & = & \{z\}
\end{array}
$$

Solution:

$$
\begin{array}{lcl}
v_1 & = & \emptyset \\
v_2 & = & \{x\} \\
v_3 & = & \{x, y\} \\
v_4 & = & \{x\} \\
v_5 & = & \{y\} \\
v_6 & = & \{z\}
\end{array}
$$

We compute the solution by going through the program backwards, from v_6 to v_1.

Is there any mathematical theory supporting this type of analysis?

Partially ordered sets

Binary relations

Let A be a set.

A binary relation $R \subseteq A \times A$ is called:

- reflexive: $(x, x) \in R$ for any $x \in A$
- symmetric: $(x, y) \in R$ implies $(y, x) \in R$ for any $x,\ y \in A$
- antisymmetric: $(x, y) \in R$ and $(y, x) \in R$ implies $x = y$ for any $x,\ y \in A$
- transitive: $(x, y) \in R$ and $(y, z) \in R$ implies $(x, z) \in R$ for any $x,\ y,\ z \in A$
- preorder: reflexive, transitive
- partial order: reflexive, antisymmetric, transitive
- equivalence: reflexive, symmetric, transitive

Partially ordered sets

A partially ordered set (**poset**) is a pair (A, R), where A is a set and R is a partial order on A.

Example.

- $(\mathcal{P}(T), \subseteq)$, T an arbitrary set,
- $(\mathbb{N}, \leq)$, $(\mathbb{N}, |)$ where $|$ is the divisibility relation,
- $(Pf(A, B), \prec)$ where

A, B are sets,
$Pf(A, B) = \{f : A \rightharpoonup B | f \text{ partial function}\}$,
$f \prec g \Longrightarrow dom(f) \subseteq dom(g)$ and $f(a) = g(a)$ for any $a \in dom(f)$

- If $(A, \leq)$ is a poset and $\geq := \leq^{-1}$ then $(A, \geq)$ is also a poset.

Totally orderd sets

Let $(A, \leq)$ be a poset. Two elements a_1, $a_2 \in A$ are **comparable** if $a_1 \leq a_2$ or $a_2 \leq a_1$.

Example. In $(\mathbb{N}, |)$ the elements 2 and 4 are comparable, but the elements 3 and 7 are not.

A partial order is called **total** (**liniar**) if any two elements are comparable.

A **totally ordered set** (or a **chain**) is a pair $(A, \leq)$ where A is a set and $\leq$ is a total order on A.

Example. $(\mathcal{LR}, \leq_{lex})$ is a chain, where $\mathcal{LR}$ is the set of words of a dictionary and $\leq_{lex}$ is the lexicographic order.

The cartesian product of chains

Assume $(A_1, \leq_1)$, $(A_2, \leq_2)$ are chains.

- On $A_1 \times A_2$ we define the **componentwise order** as follows:
  $(x_1, x_2) \leq (y_1, y_2) \Leftrightarrow x_1 \leq_1 y_1$ and $x_2 \leq_2 y_2$.
  Then $(A_1 \times A_2, \leq)$ is a poset.
- If $|A_1|, |A_2| \geq 2$ then $(A_1 \times A_2, \leq)$ is **not** a chain.
- On $A_1 \times A_2$ we define the **lexicographic order** by
  $(x_1, x_2) \leq_{lex} (y_1, y_2) \Leftrightarrow (x_1 \leq_1 y_1$ si $x_1 \neq y_1)$ or
  $(x_1 = y_1$ si $x_2 \leq_2 y_2)$.
  Then $(A_1 \times A_2, \leq)$ is a chain.

Exercise. Define the lexicographic order on the cartesian product of a finite collection of chains $(A_1, \leq_1), \cdots, (A_n, \leq_n)$.

Minimal and maximal elements

Let $(A, \leq)$ be a **poset**. An element $e \in A$ is called:

- **minimal element** if $(a \leq e \Rightarrow a = e)$;
- **the least (bottom, minimum) element** if $e \leq a$ or $a \in A$;
- **maximal element** if $(e \leq a \Rightarrow a = e)$;
- **the greatest (top, maximum) element** if $a \leq e$ or $a \in A$.

Example. In the poset $(\{2, 4, 5, 10, 12, 20, 25\}, |)$, the minimal elements are 2 si 5, and the maximal elements are 12, 20, 25.

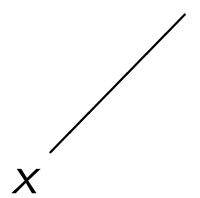
Remark. The least (greatest) element is a minimal (maximal) element. The converse is not true!

Hasse Diagram

In the poset $(A, \leq)$ the *covering relation* is defined by

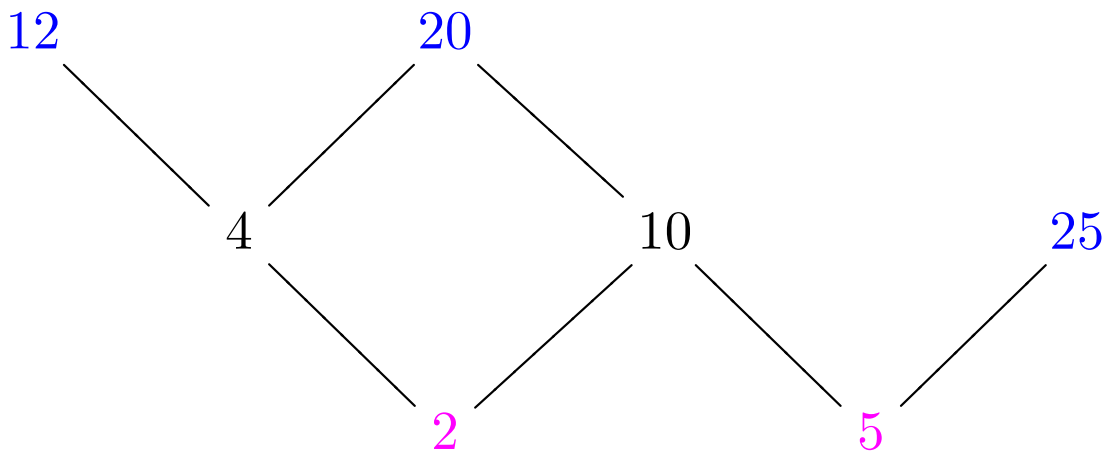
$a \prec b$ if $\quad a \leq b$ and $a \neq b$ and

$a \leq x \leq b$ implies $x = a$ or $x = b$ for any $x \in A$

Representing $a \prec b$ by 

the we get the *Hasse diagram* of the given poset.

Example. The Hasse diagram of the poset $(\{2, 4, 5, 10, 12, 20, 25\}, |)$ is



Infimum and supremum

Let $(A, \leq)$ be a poset and $X \subseteq A$.
An element $a \in A$ is called

- **lower bound** for X if $a \leq x$ for any $x \in X$;
- **upper bound** for X if $x \leq a$ for any $x \in X$;
- **infimum** for X if a is the greatest lower bound; we denote $a = \inf X$;
- **supremum** for X if a is the lowest upper bound; we denote $a = \sup X$;

Exercise. The infimum (supremum) of a set, if it exists, is unique.

Examples

- $(\mathbb{R}, \leq)$, $X = (3, 4]$, $Y = \mathbb{N}$

(a) the set of all the upper bounds for X is $[4, \infty)$,
(b) 4 is the supremum of X,
(c) the set of all the lower bounds for X is $(-\infty, 3]$,
(d) 3 is the infimum of X,
(e) Y has no upper bounds,
(f) the set of all the lower bounds for Y is $(-\infty, 0)$,
(g) 0 is the infimum of Y .

- $(\mathbb{N}, |)$, $X = \{n_1, n_2\}$

(a) $\sup X = \sup\{n_1, n_2\} = lcm\{n_1, n_2\}$,
(b) $\inf X = \inf\{n_1, n_2\} = gcd\{n_1, n_2\}$.

- $(\{2, 4, 5, 10, 12, 20, 25\}, |)$, $X = \{12, 20, 25\}$

X has no lower or upper bounds.

- $(\mathcal{P}(T), \subseteq)$, $\mathcal{X} \subseteq \mathcal{P}(T)$

$\sup \mathcal{X} = \bigcup\{Y | Y \in \mathcal{X}\}$, $\inf \mathcal{X} = \bigcap\{Y | Y \in \mathcal{X}\}$

Lattices

Lattices

Definitia L1.
A poset $(L, \leq)$ is a *lattice* if $\sup\{x_1, x_2\}$, $\inf\{x_1, x_2\}$ exists for any two elements x_1, $x_2 \in L$.

The infimum and the supremum become *operations* on L:
$\vee : L \times L \to L$, $x_1 \vee x_2 := \sup\{x_1, x_2\}$,
$\wedge : L \times L \to L$, $x_1 \wedge x_2 := \inf\{x_1, x_2\}$.

Proposition

The following identities hold:

- associativity:
  $(x \vee y) \vee z = x \vee (y \vee z)$, $(x \wedge y) \wedge z = x \wedge (y \wedge z)$,
- commutativity: $x \vee y = y \vee x$, $x \wedge y = y \wedge x$,
- absorption: $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$.

A lattice $(L, \leq)$ is an algebraic structure $(L, \vee, \wedge)$.

Bounded lattices

A lattice is called *bounded* if it has greatest element 1 and lowest element 0.

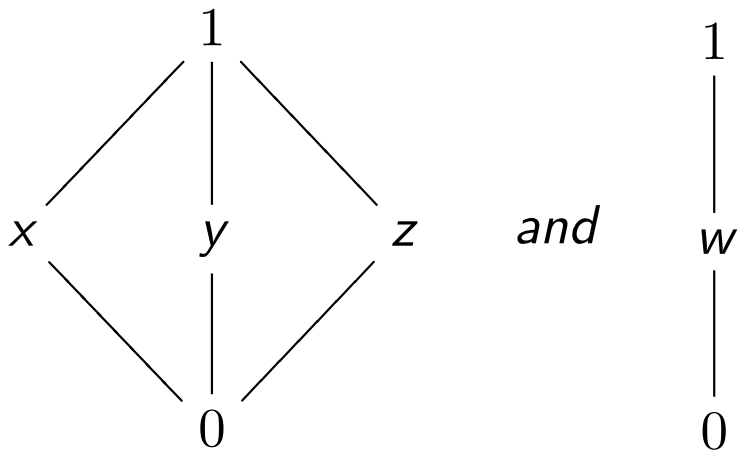
A bounded lattice will be denoted by $(L, \leq, 0, 1)$ (as a poset), and by $(L, \vee, \wedge, 0, 1)$ as algebraic structure. Note that

$$x \vee 0 = x,\ x \wedge 0 = 0,\ x \vee 1 = 1,\ x \wedge 1 = x \text{ for any } x \in L.$$

An element x is *a complement* of y if $x \vee y = 1$ si $x \wedge y = 0$.

A lattice is *complemented* if any element has a unique complement.

Example. Find the complemented elements of



Boolean algebra

A lattice $(L, \vee, \wedge)$ is *distributive* if, for any x, $y \in L$:

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \text{ and } x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

Property

In a bounded and distributive lattice, any element has at most one complement.

Boolean Algebra

A *Boolean algebra* is a bounded, distributive and complemented lattice.

Boolean algebras provide an algebraic approach to logical reasoning.

Fixed point theorems

CPO

CPO

A *complete partial order* (CPO) is a poset $(C, \leq, \bot)$ such that:

- C has a bottom element $\bot$,
- $\sup X$ exists for any *chain* $X \subseteq C$.

Example. $(Pf(A, B), \prec, \bot)$ is a CPO, where $\bot$ is the empty (nowhere defined) function.

Complete lattice

A lattice $(L, \leq)$ is *complete* if $\inf X$ and $\sup X$ exist for any $X \subseteq L$.

Example. $(\mathcal{P}(A), \subseteq)$ is a complete lattice.

Remark

(a) Any complete lattice is a CPO.
(b) Any complete lattice is bounded.

Monotone functions. Fixed points.

Monotone function

Given two posets $(A, \leq_A)$ and $(B, \leq_B)$, we say that a function $f : A \to B$ is *monotone* (increasing) if $a_1 \leq_A a_2 \Rightarrow f(a_1) \leq_B f(a_2)$ for any $a_1, a_2 \in A$.

Continuous function

Given two CPOs $(A, \leq_A)$ si $(B, \leq_B)$ we say that a function $f : A \to B$ is *continuous* if

$$f(\sup\{a_n | n \in \mathbb{N}\}) = \sup\{f(a_n) | n \in N\}$$

for any chain $\{a_n | n \in \mathbb{N}\}$ din A.

Remark. Any continuous function is monotone.

Fixed point theorems

Fixed point

An element $a \in A$ is a *fixed point* of $f : A \to A$ if $f(a) = a$.

Fixed point theorem for complete lattices (Knaster-Tarski)

If $(L, \leq)$ is a complete lattice and $\mathbf{F} : L \to L$ is a monotone (increasing) function, then $a = \inf\{x \in L | \mathbf{F}(x) \leq x\}$ is the least fixed point of $\mathbf{F}$.

Fixed point theorem for CPOs (Kleene)

If $(C, \leq, \bot)$ is a CPO and $\mathbf{F} : C \to C$ is a continuous function, then $a = \sup\{\mathbf{F}^n(\bot) | n \in \mathbb{N}\}$ is the least fixed point of $\mathbf{F}$.

(we note that a exists since the sequence $\mathbf{F}^0(\bot) = \bot \leq \mathbf{F}(\bot) \leq \mathbf{F}^2(\bot) \leq \cdots \leq \mathbf{F}^n(\bot) \leq \cdots$ is a chain.)

Fixed point theorem

Example.

$Pf(\mathbb{N},\mathbb{N})$ CPO, $\perp : \mathbb{N} \rightharpoonup \mathbb{N}$, $\perp(k)$ undefined for any $k \in \mathbb{N}$

$\mathbf{F} : Pf(\mathbb{N},\mathbb{N}) \to Pf(\mathbb{N},\mathbb{N})$

$$\mathbf{F}(g)(k) := \begin{cases} 1, & k = 0 \\ k * g(k-1), & k > 0 \text{ and } g(k-1) \text{is defined} \\ \text{undefined}, & \text{otherwise} \end{cases}$$

Since $\mathbf{F}$ is continuous, the least fixed point of $\mathbf{F}$ is $f = \sup\{\mathbf{F}^n(\perp) | n \in \mathbb{N}\}$.

$$f(k) = \mathbf{F}(f)(k) := \begin{cases} 1, & k = 0 \\ k * f(k-1), & k > 0 \text{ si } f(k-1) \text{ is defined} \\ \text{undefined}, & \text{otherwise} \end{cases}$$

f is the *factorial function*.

Fixed point theorem for CPOs: proof

Assume $(C, \leq)$ is a CPO and $\mathbf{F} : C \to C$ is a continuous function. We have to prove that $a = \sup\{\mathbf{F}^n(\bot) | n \in \mathbb{N}\}$ is the least fixed point of $\mathbf{F}$.

(I) $a = \sup\{\mathbf{F}^n(\bot) | n \in \mathbb{N}\}$ is a fixed point

$$\begin{aligned}\mathbf{F}(a) &= \mathbf{F}(\sup\{\mathbf{F}^n(\bot) | n \in \mathbb{N}\}) \\ &= \sup\{\mathbf{F}(\mathbf{F}^n(\bot)) | n \in \mathbb{N}\} \text{ (continuitate)} \\ &= \sup\{\mathbf{F}^{n+1}(\bot) | n \in \mathbb{N}\} \\ &= \sup\{\mathbf{F}^n(\bot) | n \in \mathbb{N}\} = a\end{aligned}$$

(II) a is the least fixed point

Assume b is another fixed point, i.e. $\mathbf{F}(b) = b$. We prove by induction on $n \geq 1$ that $\mathbf{F}^n(\bot) \leq b$.

For $n = 0$, $\mathbf{F}^0(\bot) = \bot \leq b$ because $\bot$ is the first element.

If $\mathbf{F}^n(\bot) \leq b$, then $\mathbf{F}^{n+1}(\bot) \leq \mathbf{F}(b)$ beacause $\mathbf{F}$ is increasind. Since $\mathbf{F}(b) = b$ we get $\mathbf{F}^{n+1}(\bot) \leq b$.

Fixed point theorem for complete lattices: proof

Assume $(L, \leq)$ is a complete lattice and $\mathbf{F} : C \to C$ is a monotone function.

(I) $a = \inf\{x \in L | \mathbf{F}(x) \leq x\}$ is a fixed point

If $X := \{x \in L | \mathbf{F}(x) \leq x\}$ then $a \leq x$ for any $x \in X$, so $\mathbf{F}(a) \leq \mathbf{F}(x) \leq x$. Consequently, $\mathbf{F}(a)$ is a lower bound for X, which implies that $\mathbf{F}(a) \leq a$. It follows that $\mathbf{F}(a) \in X$, which implies that $a \leq \mathbf{F}(a)$. We proved that $\mathbf{F}(a) \leq a$ and $a \leq \mathbf{F}(a)$, so $\mathbf{F}(a) = a$.

(II) a is the least fixed point Assume b is another fixed point, i.e. $\mathbf{F}(b) = b$. Then

$b \in X$, so $a \leq b$.

Static Analysis: Flow-sensitive analysis

Back to our problem

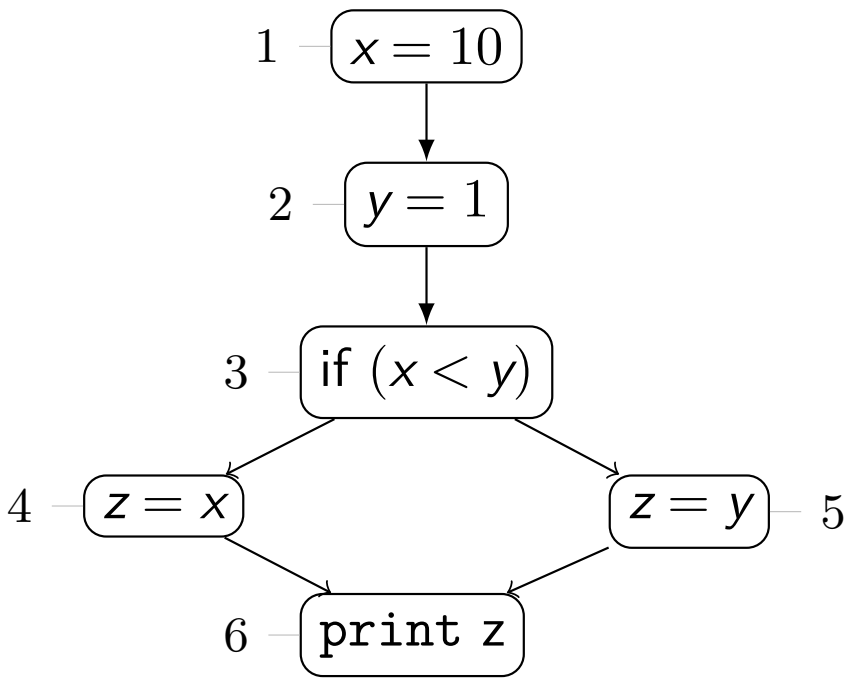
$$v_n = \left(\bigcup\{v_i \mid i \text{ succesor of } n\} \setminus \mathit{Written}(n)\right) \cup \mathit{Read}(n)$$

Constraints:

$$\begin{array}{lcl} v_1 & = & v_2 \setminus \{x\} \\ v_2 & = & v_3 \setminus \{y\} \\ v_3 & = & v_4 \cup v_5 \cup \{x, y\} \\ v_4 & = & (v_6 \setminus \{z\}) \cup \{x\} \\ v_5 & = & (v_6 \setminus \{z\}) \cup \{y\} \\ v_6 & = & \{z\} \end{array}$$

Solution:

$$\begin{array}{lcl} v_1 & = & \emptyset \\ v_2 & = & \{x\} \\ v_3 & = & \{x, y\} \\ v_4 & = & \{x\} \\ v_5 & = & \{y\} \\ v_6 & = & \{z\} \end{array}$$



Remarks

In our previous solution $v_1, \cdots, v_n \in \mathcal{P}(Var)$, where Var is the set of variables.

If we denote $L = \mathcal{P}(Var)$ then our system of constraints can be written as follows:

$$\begin{array}{lcl} x_1 & = & F_1(x_1, \ldots, x_n) \\ x_2 & = & F_2(x_1, \ldots, x_n) \\ & \ldots & \\ x_n & = & F_n(x_1, \ldots, x_n) \end{array}$$

where $x_1, \ldots, x_n \in L$ and $F_1, \ldots, F_n : L^n \to L$.

Moreover, if we define $F : L^n \to L^n$ by

$$F(x_1, \ldots, x_n) = (F_1(x_1, \ldots, x_n), \ldots, F_n(x_1, \ldots, x_n))$$

then our system is equivalent to

$$F(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$$

We want to use a fixed point theorem!

The product lattice

Let $(L, \leq)$ be a lattice and $n \geq 1$.

The product lattice L^n

On L^n we consider the componentwise order:

$$(x_1, \ldots, x_n) \leq_n (y_1, \ldots, y_n) \text{ ddacă } x_i \leq y_i \text{ pt. orice } i \in \{1, \ldots, n\}$$

Hence $(L^n, \leq_n)$ is a lattice. Moreover, if $(L, \leq)$ is complete, then $(L^n, \leq_n)$ is also complete.

Applying the fixed point theorem on complete lattices we get:

If $(L, \leq)$ is a complete lattice and $F_1, \ldots, F_n : L^n \to L$ are monotone functions then the function $F : L^n \to L^n$ defined by

$$F(x_1, \ldots, x_n) = (F_1(x_1, \ldots, x_n), \ldots, F_n(x_1, \ldots, x_n))$$

has a least fixed point. If, in addition, $F_1, \ldots, F_n$ are continuous, the least fixed point is $\sup\{F^n(\bot, \ldots, \bot) | n \in \mathbb{N}\}$ where $\bot$ is the first element of L.

Live variable analysis

Using the previous setting, in order to solve our system

$$\begin{array}{lcl} v_1 & = & v_2 \setminus \{x\} \\ v_2 & = & v_3 \setminus \{y\} \\ v_3 & = & v_4 \cup v_5 \cup \{x, y\} \\ v_4 & = & (v_6 \setminus \{z\}) \cup \{x\} \\ v_5 & = & (v_6 \setminus \{z\}) \cup \{y\} \\ v_6 & = & \{z\} \end{array}$$

we have to find the least fixed point of

$$F(v_1, v_2, v_3, v_4, v_5, v_6) =$$
$$(v_2 \setminus \{x\}\,,\ v_3 \setminus \{y\}\,,\ v_4 \cup v_5 \cup \{x, y\}\,,\ (v_6 \setminus \{z\}) \cup \{x\}\,,\ (v_6 \setminus \{z\}) \cup \{y\}\,,\ \{z\})$$

in the complete lattice $(\mathcal{P}(Var), \subseteq, \emptyset, Var)$.

Live variable analysis

$F(v_1, v_2, v_3, v_4, v_5, v_6) =$
$(v_2 \setminus \{x\}\,,\ v_3 \setminus \{y\}\,,\ v_4 \cup v_5 \cup \{x, y\}\,,\ (v_6 \setminus \{z\}) \cup \{x\}\,,\ (v_6 \setminus \{z\}) \cup \{y\}\,,\ \{z\})$

Using Kleene's theorem, we find the fixed point as follows:

	v_1	v_2	v_3	v_4	v_5	v_6
$\bot$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$F(\bot)$	$\emptyset$	$\emptyset$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^2(\bot)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^3(\bot)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$

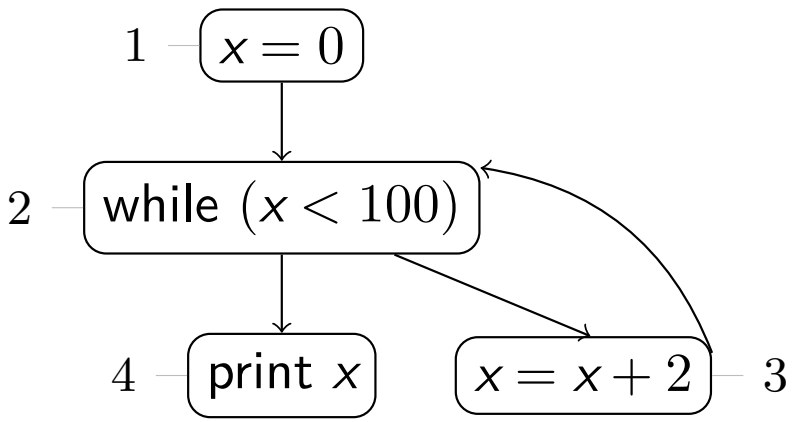
where $\bot = (\emptyset, \ldots, \emptyset)$

Static Analysis: Abstract interpretations

CFG

The Control Flow Graph allows us to visualize the possible executions of a program.

```
x = 0;
while (x<100)
      {x=x+2;}
```



What about *the meaning* of a given program?

Standard semantics

Semantics describe formally what a program means.

Starting with the CFG we define a *standard semantics* in terms of *transition functions* between pairs (σ, k) where σ is a *state* and k is a node in the CFG.

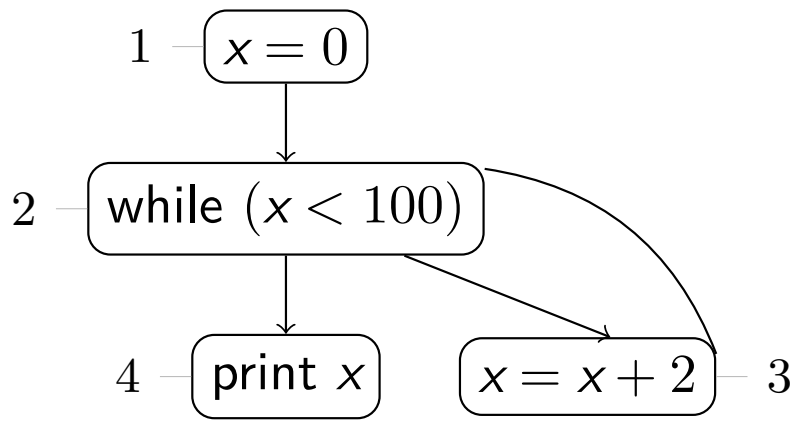
$\mathit{Nodes} = \{1, 2, 3, 4\}$,

What is a state?
A *state* is a (partial) function from variables to values.

$\mathit{Var} = \{x\}$,
$\mathit{St} = \mathbb{Z}^{\mathit{Var}} \simeq \mathbb{Z}$ (memory states)

$\mathit{LSt} = \mathit{St} \times \mathit{Nodes}$ (labelled states)

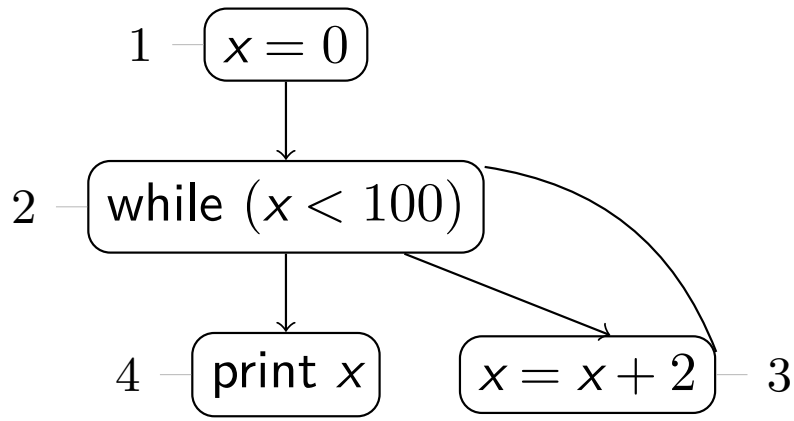
$\tau : \mathit{LSt} \to \mathit{LSt}$
$\tau(x, 1) = (0, 2)$
$\tau(x, 2) = (x, 3)$ if $x < 100$
$\tau(x, 2) = (x, 4)$ if $x \geq 100$
$\tau(x, 3) = (x + 2, 2)$
$\tau(x, 4) = (x, 4)$



Collective semantics

$Nodes = \{1, 2, 3, 4\}$, $Var = \{x\}$,
$St = \mathbb{Z}^{Var} \simeq \mathbb{Z}$
$LSt = St \times Nodes$

$\tau : \mathcal{P}(LSt) \to \mathcal{P}(LSt)$
$\tau(\{(x, 1)\}) = \{(0, 2)\}$
$\tau(\{(x, 2)\}) = \{(x, 3)\}$ if $x < 100$
$\tau(\{(x, 2)\}) = \{(x, 4)\}$ if $x \geq 100$
$\tau(\{(x, 3)\}) = \{(x + 2, 2)\}$
$\tau(\{(x, 4)\}) = \{(x, 4)\}$



$\tau(S) = \bigcup_{s \in S} \tau(s)$

For $I \subseteq LSt$ we define
$\Phi_I : \mathcal{P}(LSt) \to \mathcal{P}(LSt)$ prin $\Phi_I(S) = I \cup \tau(S)$

Using The fixed point theorem we can calculate $R = lfp(\Phi_I)$, the least fixed point of Φ_I.

What is R?
R is the set of reachable states from the initial set I.

Abstract interpretation - general idea

The collective semantics is a *concrete semantics*, that defines the meaning of a program as a (transition) function between labelled states.

We would like to analyze the program with emphasis on a particular property of states. This process is called abtraction and it is represented by a function

$$\alpha : \mathcal{P}(States) \to A$$

where *States* is the set of states and A is the abstract domain corresponding to the particular property we chose.

The reverse process is called concretization and it is represented by a function

$$\gamma : A \to \mathcal{P}(States)$$

Moreover, we assume that α and γ define a Galois connection:

$$\mathcal{P}(States) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$$

Galois connection

Assume $(C, \leq_C)$ and $(A, \leq_A)$ are posets. A *Galois connection* is defined by a pair of functions

$$C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$$

that satisfy the following property:

$$\alpha(c) \leq_A a \text{ if and only if } c \leq_C \gamma(a)$$

for any $a \in A$ and $c \in C$.

Properties:

- $c \leq \gamma(\alpha(c))$ for $c \in C$ and $\alpha(\gamma(a)) \leq a$ for $a \in A$;
- α and γ are increasing;
- $a \in \alpha(C)$ iff a is a fixed point for $\alpha \circ \gamma$;
- $c \in \gamma(A)$ iff c is a fixed point for $\gamma \circ \alpha$.

Example: abstract interpretation

The abstract domain is a lattice $(A = \{pos, neg, zero, \bot, \top\}, \leq)$ where the order relation is

$$\bot \leq a \leq \top \text{ for any } a \in \{pos, neg, zero\}.$$

Let $LSt = \mathbb{Z} \times Nodes$ from our previous example. We define

- abstraction: $\alpha : \mathcal{P}(LSt) \to A$
  $\alpha(\emptyset) = \bot$
  $\alpha(S) = pos$ if $x > 0$ for any $(x, n) \in S$
  $\alpha(S) = zero$ if $x = 0$ for any $(x, n) \in S$
  $\alpha(S) = neg$ if $x < 0$ for any $(x, n) \in S$
  $\alpha(S) = \top$ otherwise.
- concretization: $\gamma : A \to \mathcal{P}(LSt)$
  $\gamma(\bot) = \emptyset$, $\gamma(\top) = LSt$,
  $\gamma(pos) = \{(x, n) \in LSt \mid x > 0\}$,
  $\gamma(neg) = \{(x, n) \in LSt \mid x < 0\}$,
  $\gamma(zero) = \{(0, n) \mid n \in Nodes\}$.

$$\mathcal{P}(St) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$$

Abstract transformation

Previously we determined the reachable states as fixed point of a function $\Phi_I : \mathcal{P}(LSt) \to \mathcal{P}(LSt)$.
Using the Galois connection

$$\mathcal{P}(LSt) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$$

we get the abstract transformation $\Phi_I^\# : A \to A$ defined by $\Phi_I^\# = \alpha \circ \Phi_I \circ \gamma$

If A is a complete lattice, then we can apply the fixed point theorem for $\Phi_I^\#$.
Moreover, the analysis is *sound*, i.e.

$$\alpha(\mathit{lfp}(\Phi_I)) \leq \mathit{lfp}(\Phi_I^\#).$$

If $\mathit{lfp}(\Phi_I^\#) \leq \alpha(\mathit{lfp}(\Phi_I))$ the analysis is *complete*.

- In the concrete semantics $\mathit{lfp}(\Phi_I)$ determines the reachable states from a given initial state.
- Consequently the abstraction of a a concrete value (a concrete set of states) is over-approximated by the result of the (abstract) analysis (an element of the sign lattice in our example).

Exercises

Exercise 1

Compute liveness information for the following program:

```
x1 = 1;
x2 = x1 + x1;
x3 = x2 + x1;
x4 = x1 +x2;
x5 =  x4+ x3;
print x5
```

Exercise 2

Prove that any finite lattice is complete

Exercise 3

Prove that the functions $F_1, \ldots, F_n$ from our example are continuous.

Exercises

Exercise 4

Prove that $Pf(A, B)$ is a CPO for any two sets A and B.

Exercise 5

Compute the fixed point R for Φ_I with $I_1 = \{(99, 2)\}$ and $I_2 = (0, 2)$.

Exercise 6

Check that $\alpha(lfp(\Phi_I)) \leq lfp(\Phi_I^{\#})$ for $I = \{(99, 2)\}$.

Exercises

Exercise 7

Let $\mathcal{L}$ be a logic, $Form_{\mathcal{L}}$ the set of its formulas, $Mod_{\mathcal{L}}$ the set of its models and $\models \subseteq Mod_{\mathcal{L}} \times Form_{\mathcal{L}}$ the satisfaction relation.

(a) Give a concrete example.

(b) Define a Galois connection between the posets

$$(\mathcal{P}(Form_{\mathcal{L}}), \subseteq) \text{ and } (\mathcal{P}(Mod_{\mathcal{L}}), \supseteq)$$

Thank you!

References for static analysis

1 Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, USA, January 1979, pages 269–282. ACM, 1979.

2 A. Moller, M. I. Schwartzbach, Static Program Analysis, `https://cs.au.dk/~amoeller/spa/`