Program Analysis - Abstract Interpretation - Dataflow Analysis

[Cousot, Cousot 1977] [Kam, Ullman 1977]

Definition

Informally: prove properties of the runtime behaviour of a program without running it

runtime behaviour: $sp(\mathbf{true}, P)$ (at every program point)

"without running it": finite time

Presburger arithmetic vs. real programs

loops

Obstacles to computability:

1. unknown input/unbounded number of inputs
2. unbounded execution paths
3. unbounded program state

Applications

1. optimizing compilers
2. development tools
3. verification and bug finding

Example program

```
n = input
if( n < 0 ) {
   a = 2
   b = 3
} else {
   a = 1
   b = 4
}
c = a + b
output(c)
```

Control flow graph

Annotate CFG with $n = ?, a = 2, b = 3, a \in \{2, 3\}$, etc.

We get $c \in \{4, 5, 6\}$, even though we know $c = 5$.

Let $\hat{s}$ be some abstraction of the program state, and let $f_{\text{stmt}}(\hat{s})$ be the state after stmt executes.

MOP (merge over all paths) = $\sqcup_{p \in path} f_p(\hat{i})$

So $f(\text{left path})$ gives $c = 5$ and $f(\text{right path})$ gives $c = 5$. So MOP gives $c = 5$. But MOP uncomputable in general.

Instead, compute LFP (MFP). LFP soundly approximates MOP.

LFP is like finding an invariant for each program point, such that the invariants are consistent with semantics of each statement. Start with all invariants **false**, then incrementally correct them to be consistent with statements.

Partial orders and lattices

Def: A relation $\sqsubseteq$ on a set S is a **partial order** if it is:

1. reflexive: $x \sqsubseteq x$
2. transitive: $x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$
3. antisymmetric: $x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$

Idea: Whenever x soundly approximates program state, then $x \sqsubseteq y$ means that y also soundly approximates program state.

e.g.: $c \in \{5\} \sqsubseteq c \in \{4, 5, 6\}$

Def: z is an **upper bound** of x and y if $x \sqsubseteq z$ and $y \sqsubseteq z$.

Def: z is a **least upper bound** (join, $\sqcup$) if z is an upper bound, and for all upper bounds z', $z \sqsubseteq z'$.

Question: lub for our example?

Def: a **lattice** L is a poset such that $\forall x, y \in L. x \sqcup y \in L \wedge x \sqcap y \in L$

Def: a **complete lattice** L is a poset closed under least upper bounds and greatest lower bounds. For every subset $S \subseteq L$, $\sqcup S \in L$.

Every finite lattice is complete. Why? Every lattice of finite height is complete. (Trickier to prove.)

Def: **bottom** $\bot = \sqcup\{\}. \forall x. \bot \sqsubseteq x$

Def: **top** $\top = \sqcup L. \forall x. x \sqsubseteq \top$

Example lattices (Hasse diagrams):

1. Powerset: $\mathcal{P}(a, b)$
2. Integers: not complete
3. Integers ordered by divisibility

Lattice constructions:

1. If S is a set, then $\mathcal{P}(S)$ is a lattice with $\sqsubseteq$ defined as $\subseteq$ or $\supseteq$.
2. If A, B are lattices, then $A \times B$ is a lattice with $\sqsubseteq$ defined as

$(a_1, b_1) \sqsubseteq (a_2, b_2) \iff a_1 \sqsubseteq a_2 \wedge b_1 \sqsubseteq b_2$.

3. If S is a set and L is a lattice, then the set of maps $S \to L$ is also a lattice with $\sqsubseteq$ defined as $m \sqsubseteq m' \Rightarrow \forall x.\, m(x) \sqsubseteq m'(x)$.

In our example, lattice is $\{a, b, c\} \to \mathcal{P}(\mathbb{Z})$.

A more practical example is $\{a, b, c\} \to \mathbb{Z}^{\top}_{\bot}$.

Monotone dataflow framework

Choose a lattice L to abstract program states. Decide the "meaning" of lattice elements: formally, concretization function $\gamma : L \to 2^{\mathbf{State}}$.

For each program statement, define function $f_{\text{stmt}} : L \to L$ to model the behaviour of stmt. f_{stmt} is called the transfer function or the abstract transformer. Given a concrete semantics $[\![\text{stmt}]\!] : \mathbf{State} \to \mathbf{State}$, the transformer should satisfy $\{[\![\text{stmt}]\!](c) \mid c \in \gamma(\ell)\} \subseteq \gamma(f_{\text{stmt}}(\ell))$.

The most precise abstract transformer is $\lambda \ell.\, \alpha(\{[\![\text{stmt}]\!](s) \mid s \in \gamma(\ell)\})$.

For a path $p = s_1, \ldots, s_n$ through the CFG, define $f_p = f_{s_n} \circ \cdots \circ f_{s_1}$.

Goal: find $MOP(n, x) = \sqcup_{p \text{ is a path from } s_0 \text{ to } s_n} f_p(x)$.

Instead, for each statement s in the CFG, find (least) $V_{s_{in}}$ and $V_{s_{out}}$ satisfying:

1. $V_{s_{out}} = f_s(V_{s_{in}})$
2. $V_{s_{in}} = \sqcup_{s' \in \text{pred}(s)} V_{s'_{out}}$

Property: $MOP(n, x) \sqsubseteq V_{n_{out}}$

Example: Constant propagation transfer function for statement a = b + c:

	$\top$	k_1	$\bot$
$\top$	$\top$	$\top$	$\top$
k_2	$\top$	$k_1 + k_2$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$

Fixed points

Def: x is a **fixed point** of f if $f(x) = x$.

Let $\mathcal{V} : L^{2n}$ (where L^{2n} is the product of L with itself $2n$ times, where n is the number of statements in the program). Define $\mathcal{V} = (V_{s_{1out}}, V_{s_{1in}}, \ldots, V_{s_{1out}}, V_{s_{1in}})$. Define $F : L^{2n} \to L^{2n}$ following the equations above. Then we seek a (least) fixed point of F so that $F(\mathcal{V}) = \mathcal{V}$.

Def: f is **monotone** if $x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$. (order-preserving)

Thm: **(Knaster-Tarski)** If L is a complete lattice and $f : L \to L$ is monotone, then f has a fixed point, and the set of all its fixed points forms a complete sub-lattice in L. In particular, f has a least fixed point. But how do we compute it?

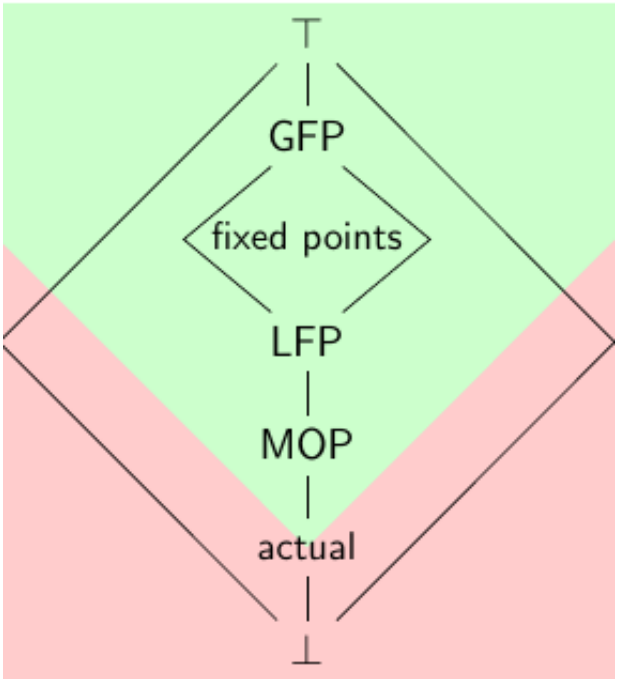
Algorithm: Compute $f^{(n)}(\bot) = f(f(\cdots(f(\bot))))$ for increasing n. Properties:

1. $\forall n. f^{(n+1)}(\bot) \sqsupseteq f^{(n)}(\bot)$.
2. If $f^{(n+1)}(\bot) = f^{(n)}(\bot)$ for some n, then $f^{(n)}(\bot)$ is a fixed point of f.
3. If L is of finite height, then n always exists (algorithm terminates) and $f^{(n)}(\bot)$ is the *least* fixed point.
4. **(Kleene)** If algorithm terminates and f is *continuous*, then $f^{(n)}(\bot)$ is the *least* fixed point. Almost every practical monotone function is continuous.

Dataflow analysis: finite height. Abstract interpretation: possibly infinite height.

Examples:

1. Constant propagation lattice: finite height.
2. Integer powerset lattice: infinite height.
3. Intervals: infinite height.



Thm: $MOP \sqsubseteq LFP$.

Thm: If f is distributive, then $MOP = LFP$.

Def: If f is monotone, then $f(x) \sqcup f(y) \sqsubseteq f(x \sqcup y)$. f is **distributive** if $f(x) \sqcup f(y) = f(x \sqcup y)$.

Examples:

1. constant propagation is not distributive.
2. copy constant propagation is distributive.
3. kill-gen problems are distributive: $f_s(x) = (x \setminus \text{kill}_s) \cup \text{gen}_s$ (where kill_s and gen_s are sets independent of s)

Chaotic iteration

We define F on a vector of V's using the equations for V_s involving f_s. Iterating F is expensive. Cheaper to use

a worklist of V's that have changed, and only update f_s to refect changes. Computes same result (folklore, proof of special case in [Kam, Ullman 1977], general proof in [Cousot, Cousot 1992]).

initialize $V_{s_{in}}$ and $V_{s_{out}}$ to $\perp$ for all s
set $V_{\text{entry}_{in}}$ to approximation of input state
add all statements s to worklist
while worklist not empty
  remove some s from worklist
  set $V_{s_{in}} = \sqcup_{s' \in \text{pred}(s)} V_{s'_{out}}$
  set $V_{s_{out}} = f_s(V_{s_{in}})$
  if $V_{s_{out}}$ has changed
    add successors of s to worklist
  end if
end while

Do on example program.

Widening

What if Kleene sequence does not terminate (lattice infinite height)? e.g. intervals

Def: ∇ is a widening operator if $x \sqcup y \sqsubseteq x \nabla y$ and for any sequence $a_i = f^{(i)}(\perp)$, the sequence $b_i = b_{i-1} \nabla a_i$ eventually reaches a fixed point.

Example for intervals: $i_1 \nabla i_2 = \begin{cases} i_1 \sqcup i_2 & \text{if } i_1 \sqsubseteq [-5, 5] \\ [-\infty, \infty] & \text{otherwise} \end{cases}$

Define the sequence $c_i = \begin{cases} \perp & \text{if } i = 0 \\ c_{i-1} & \text{if } f(c_{i-1}) \sqsubseteq c_{i-1} \\ c_{i-1} \nabla f(c_{i-1}) & \text{otherwise} \end{cases}$

The sequence c_i eventually reaches a fixed point. The fixed point c of c_i may not be a fixed point of f, but $f(c) \sqsubseteq c$, and $LFP(f) \sqsubseteq c$, so c is a sound approximation of $LFP(f)$.