CIS 670: Program Analysis

Title: Abstract Interpretation.

Guest Lecturer: Sriram Sankaranarayanan.

NEC Labs America,

Princeton, NJ.

`sriram@nec-labs.com`

Date: Oct 1st, 2007.

Topics

- Programs, Flowcharts, etc...
- Instances of Abstract Interpretation:
  - Analysis #1: Sign Analysis.
  - Analysis #2: Interval Analysis.
- "Concrete" Interpretation.
- Abstract Interpretation.

What is Abstract Interpretation?

Formal study of fixed points for program analysis applications.

- Program Verification Applications:
  Astrée, PolySpace, CoVerity, Absinthe, F-Soft, CodeSurfer, Fluctuat, Airac, TVLA, ...
- Denotational Semantics.
- Type Checking/Inference.

Goals

- There are numerous presentations of abstract interpretation.

- Our goal:
  1. Today: Understand the essence of the theory (without too much "Greek").
  2. Today & Wednesday: The actual theory.
  3. Wednesday: A quick guided tour through important applications & research frontiers.
- Will try to be self-contained as much as possible.

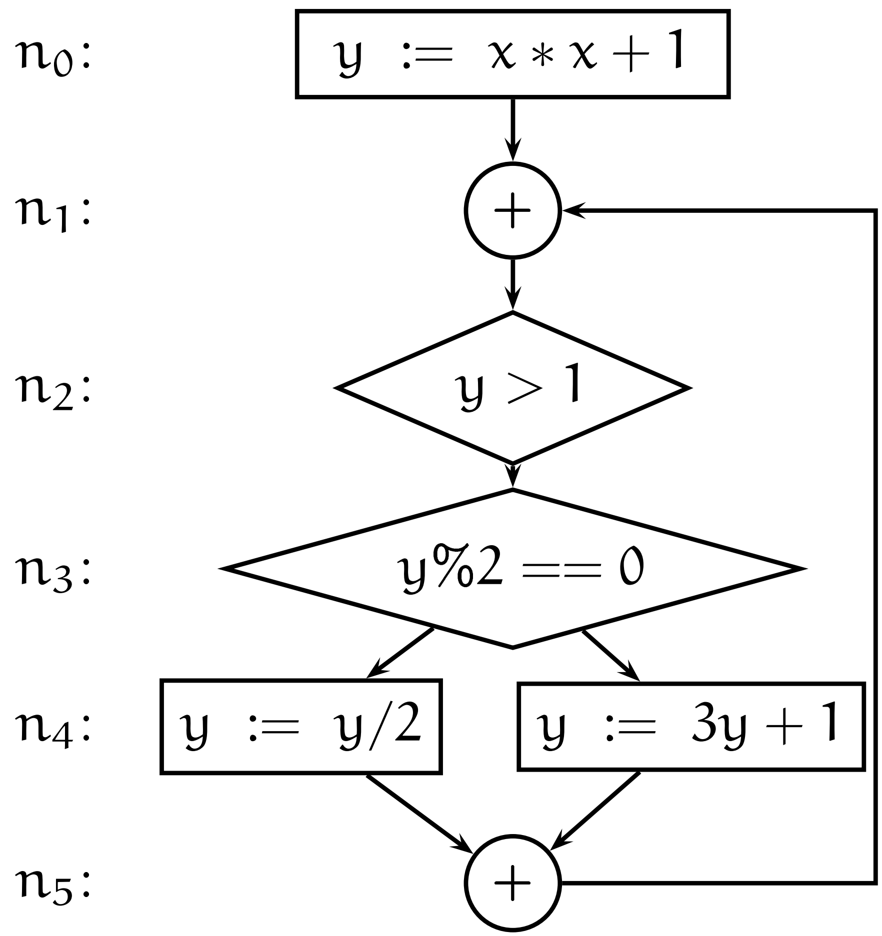
Some History

- Theory of inductive invariants: [Floyd, 1967] and [Hoare, 1969].
- Denotational Semantics: [Scott, Reynolds, Abramsky, ...]
- Invariant Generation: [King, 1969; Manna & Katz, 1975]
- Monotone frameworks for dataflow analysis: [Burstall, 1973].
- Linear equality invariant generation: [Karr, 1976]
- Interval analysis: [Cousot & Cousot, 1976]
- Abstract interpretation theory: [Cousot&Cousot, 1977]
- Polyhedral Analysis: [Cousot & Halbwachs, 1978]
- Recent advances & applications.

Programs

We will use a standard flowchart representation.

```
function foo ( int x )
int y := x * x + 1 ;
while y > 1 do
   if y%2 == 0 then
      y = y/2
   else
      y = 3 * y + 1
   end if
end while
end function
```



Program: Assumptions

For simplicity, we make the following assumptions about our programs:

- All variables are integers or reals.
- No function calls.
- No pointers, arrays, compound objects.

Note: Real program analysis tools handle features such as function calls, arrays, pointers and compound structures.

Signs Analysis

For each location n,

For each variable x,

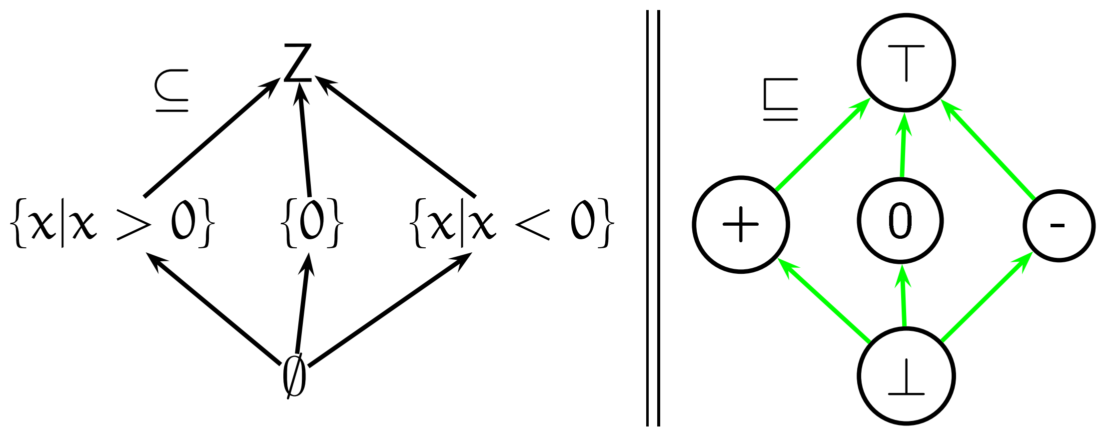
$$\mathsf{sign}(n, x) = \begin{cases} \text{``}\bot\text{''} & \text{if control never reaches } n \\ \text{``}+\text{''}, & \text{if } x > 0, \text{ whenever control reaches } n, \\ \text{``}-\text{''} & \text{if } x < 0, \cdots \\ \text{``}0\text{''} & \text{if } x = 0, \cdots \\ \text{``}\top\text{''} & \text{otherwise} \end{cases}$$

Signs Lattice

The symbols $\{\bot, +, -, 0, \top\}$ correspond to the sets of natural numbers:

$$
\begin{array}{lll|lll}
[\![\bot]\!] & : & \emptyset & [\![\top]\!] & : & \mathsf{Z} \\
[\![+]\!] & : & \{x|x > 0\} & [\![-]\!] & : & \{x|x < 0\}
\end{array}
$$

Sets of natural numbers can be ordered by inclusion $\subseteq$.



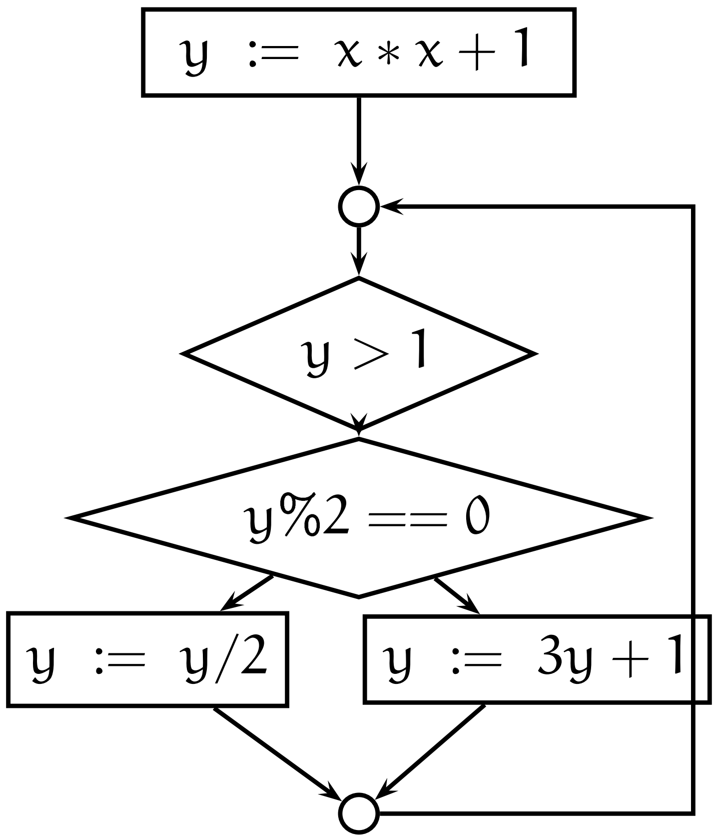
The $\subseteq$ induces a $\sqsubseteq$ relation in the sign domain.

Signs Analysis: Problem Statement

For each variable x, and each location n compute the least element of the signs lattice c such that $[\![c]\!]$ contains all the possible values of x seen when control reaches location n.

Signs Analysis: Least Solution

Assume input state $\mathsf{sign}(x, n_0) : +$, $\mathsf{sign}(y, n_0) : \top$.



Flowchart	$\mathsf{sign}(y, n)$	$\mathsf{sign}(x, n)$
n_0: $y := x * x + 1$	$\top$	$+$
n_1:	$+$	$+$
n_2: $y > 1$	$+$	$+$
n_3: $y\%2 == 0$	$+$	$+$
n_4: $y := y/2$; $y := 3y + 1$	$+$	$+$
n_5:	$+$	$+$

Signs Analysis: Problem Statement (Attempt # 2)

For each variable x, and each location n compute the ~~least element~~ some element of the signs lattice c such that $[\![c]\!]$ contains all the possible values of x seen when control reaches location n.

- Least element of the lattice cannot be computed.

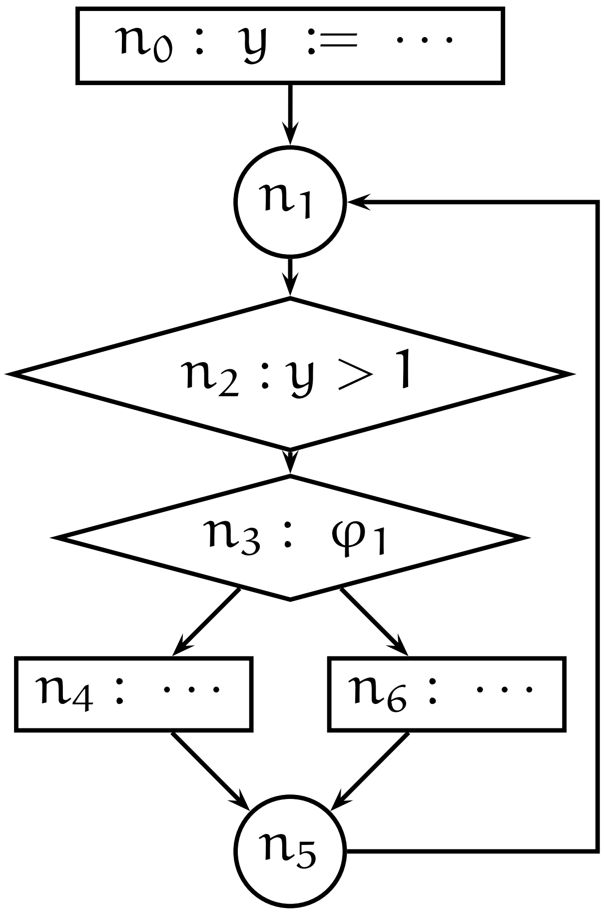
  Proof: Reduce from Hilbert's 10^{th} problem.

- Trivial analysis result: $\mathsf{sign}(n, x) : \top$ at all locations!!

- Least solution in the lattice.

  1. Derive lattice inequalities for the overapproximations.
  2. Solve them to derive a safe overapproximation.

Signs Analysis: Data-flow equations



$$\mathsf{post}(n_0, \mathsf{sign}(n_0, x)) \sqsubseteq \mathsf{sign}(n_1, y)$$

$$\mathsf{sign}(n_5, y) \sqsubseteq \mathsf{sign}(n_1, y)$$

$$\mathsf{sign}(n_1, y) \sqsubseteq \mathsf{sign}(n_2, y)$$

$$\mathsf{sign}(n_2, y) \sqcap \alpha([\![y > 1]\!]) \sqsubseteq \mathsf{sign}(n_3, y)$$

$$\mathsf{sign}(n_3, y) \sqcap \alpha([\![\varphi_1]\!]) \sqsubseteq \mathsf{sign}(n_4, y)$$

$$\mathsf{post}(n_4, \mathsf{sign}(n_4, y)) \sqsubseteq \mathsf{sign}(n_5, y)$$

$$\mathsf{post}(n_6, \mathsf{sign}(n_6, y)) \sqsubseteq \mathsf{sign}(n_5, y)$$

Abstraction

Given a set of integers I , $\alpha(I)$ is the smallest value in the sign lattice that covers it.

$$\alpha(I) \triangleq \min_{\sqsubseteq}\{c \in \mathsf{sign} | [\![c]\!] \supseteq I\}.$$

Example:

I	$\alpha(I)$
$\{x \mid x > 10\}$	$+$
$\{x \mid x <= 1\}$	$\top$
$\{x \mid x <= 0\}$	$\top$
$\{x \mid x < 0\}$	$-$
$\emptyset$	$\bot$

Signs Analysis: Post-Condition

Consider an assignment $y := \mathsf{expr}(x_1, \ldots, x_n)$.

Post: Given the sign values of $x_1, \ldots, x_n$ before an assignment, compute the sign value of y after the assignment.

$$\mathsf{post}(n : y := \mathsf{expr}, \langle \mathsf{sign}(n, x_1), \ldots, \mathsf{sign}(n, x_n) \rangle).$$

Example: Consider assignment $n_0 : y := x * x + 1$

x	$\bot$	$+$	0	$-$	$\top$
$\mathsf{post}(y := \cdots, \mathsf{sign}(n, x))$	$\bot$	$+$	$+$	$+$	$+$

Computing post condition

Goal: Compute $\mathsf{post}(y := \mathsf{expr}, \langle \mathsf{sign}(n, x_0), \ldots, \mathsf{sign}(n, x_m) \rangle)$.

Just "follow" the expression syntax.

Example: Let $\mathsf{expr} : y - z - x + 1$ and
$\langle \mathsf{sign}(n, x) : \text{“}0\text{”}, \mathsf{sign}(n, y) : \text{“}-\text{”}, \mathsf{sign}(n, z) : \text{“}+\text{”} \rangle$.

1. $p(y - z) = p(\text{“}-\text{”} - \text{“}+\text{”}) = \text{“}-\text{”}$
2. $p((y - z) - x) = p(\text{“}-\text{”} - \text{“}0\text{”}) = \text{“}-\text{”}$
3. $p(((y - z) - x) + 1) = p(\text{“}-\text{”} + \text{“}+\text{”}) = \text{“}\top\text{”}$

$\therefore\ \mathsf{post}(\mathsf{expr}, \langle \mathsf{sign}(n, x) : \text{“}0\text{”}, \mathsf{sign}(n, y) : \text{“}-\text{”}, \mathsf{sign}(n, z) : \text{“}+\text{”} \rangle) = \top$.

Post condition: Example

Assignment n_0 : $y := x * x + 1$.

x	$\bot$	$+$	0	$-$	$\top$
$post(y := \cdots, sign(n, x))$	$\bot$	$+$	$+$	$+$	$+$

Assignment n_4 : $y := 3 * y + 1$.

y	$\bot$	$+$	0	$-$	$\top$
$post(y := \cdots, sign(n, y))$	$\bot$	$+$	$+$	$-$	$\top$

Assignment n_6 : $y := y/2$.

y	$\bot$	$+$	0	$-$	$\top$
$post(y := \cdots, sign(n, y))$	$\bot$	$\top$	0	$\top$	$\top$

Lattices, Monotonic Functions & Fixed Points

Poset: A set L with a partial order $\sqsubseteq$.

Meet & Join:

$$
\begin{aligned}
a \sqcap b &= \mathsf{max}_{\sqsubseteq}\{c \mid c \sqsubseteq a, c \sqsubseteq b\} \\
a \sqcup b &= \mathsf{min}_{\sqsubseteq}\{c \mid a \sqsubseteq c, b \sqsubseteq c\}
\end{aligned}
$$

Lattice: Meets and Joins exists for every pair a, b.
(Therefore, meet and join exists every finite subset)

Complete Lattice: Every subset has a meet and a join.
(related concept: semi-complete lattice).

Monotone function: $f : L \mapsto L$, s.t. $a \sqsubseteq b \Rightarrow f(a) \sqsubseteq f(b)$.

Fixed Point: $a = f(a)$.

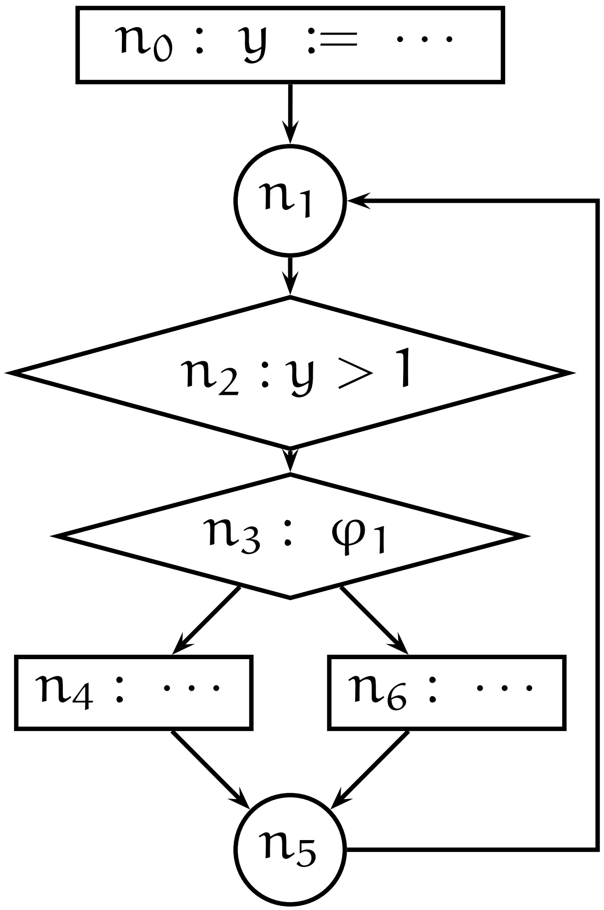
Theorem: [Knaster, 1928; Tarski, 1953]

> Every monotone function on a complete lattice has a least and greatest fixed point.

Proof:

$$\begin{aligned} \mathsf{LFP}(f) &\quad : \quad \mathsf{max}_{\sqsubseteq}(\bot, f(\bot), f^2(\bot), \ldots,) \\ \mathsf{GFP}(f) &\quad : \quad \mathsf{min}_{\sqsubseteq}(\top, f(\top), f^2(\top), \ldots,) \end{aligned}$$

Signs Analysis: Data-flow equations



$$\mathsf{post}(n_0, \mathsf{sign}(n_0, x)) \sqsubseteq \mathsf{sign}(n_1, y)$$

$$\mathsf{sign}(n_5, y) \sqsubseteq \mathsf{sign}(n_1, y)$$

$$\mathsf{sign}(n_1, y) \sqsubseteq \mathsf{sign}(n_2, y)$$

$$\mathsf{sign}(n_2, y) \sqcap \text{“}+\text{''} \sqsubseteq \mathsf{sign}(n_3, y)$$

$$\mathsf{sign}(n_3, y) \sqcap \text{“}\top\text{''} \sqsubseteq \mathsf{sign}(n_4, y)$$

$$\mathsf{post}(n_4, \mathsf{sign}(n_4, y)) \sqsubseteq \mathsf{sign}(n_5, y)$$

$$\mathsf{post}(n_6, \mathsf{sign}(n_6, y)) \sqsubseteq \mathsf{sign}(n_5, y)$$

Signs Analysis: Data-flow equations

$$
\begin{array}{rcl}
\text{“}\top\text{”} & \sqsubseteq & \mathsf{sign}(n_0, y) \\
\mathsf{post}(n_0, \mathsf{sign}(n_0, x)) \sqcup \mathsf{sign}(n_5, y) & \sqsubseteq & \mathsf{sign}(n_1, y) \\
\mathsf{sign}(n_1, y) & \sqsubseteq & \mathsf{sign}(n_2, y) \\
\mathsf{sign}(n_2, y) \sqcap \text{“}+\text{”} & \sqsubseteq & \mathsf{sign}(n_3, y) \\
\mathsf{sign}(n_3, y) \sqcap \text{“}\top\text{”} & \sqsubseteq & \mathsf{sign}(n_4, y) \\
\mathsf{post}(n_4, \mathsf{sign}(n_4, y)) \sqcup \mathsf{post}(n_6, \mathsf{sign}(n_6, y)) & \sqsubseteq & \mathsf{sign}(n_5, y) \\
\hline
\text{“}+\text{”} & \sqsubseteq & \mathsf{sign}(n_0, x) \\
\mathsf{sign}(n_0, x) \sqcup \mathsf{sign}(n_5, x) & \sqsubseteq & \mathsf{sign}(n_1, x) \\
\mathsf{sign}(n_1, x) & \sqsubseteq & \mathsf{sign}(n_2, x) \\
 & \vdots & \\
\mathsf{sign}(n_4, x) \sqcup \mathsf{sign}(n_6, x) & \sqsubseteq & \mathsf{sign}(n5, x)
\end{array}
$$

Shorthand notation: $x_i : \mathsf{sign}(n_i, x)$ and $y_i : \mathsf{sign}(n_i, y)$.

We may write the (in)equalities as :

$$
\begin{aligned}
f_1(x_1, \ldots, x_n, y_1, \ldots, y_n) &\sqsubseteq y_1 \\
&\vdots \\
f_n(x_1, \ldots, x_n, y_1, \ldots, y_n) &\sqsubseteq y_n \\
g_1(x_1, \ldots, x_n, y_1, \ldots, y_n) &\sqsubseteq x_1 \\
&\vdots \\
g_n(x_1, \ldots, x_n, y_1, \ldots, y_n) &\sqsubseteq x_n
\end{aligned}
$$

Inequations over a lattice

Let L be a lattice and $f_1, \ldots, f_n$ be monotonic functions:

$$f_j : L \times \cdots \times L \mapsto L \, .$$

Corollary **Knaster-Tarski Theorem**: The inequality system

$$f_1(x_1, \ldots, x_n) \sqsubseteq x_1, f_2(x_1, \ldots, x_n) \sqsubseteq x_2, \cdots, f_n(x_1, \ldots, x_n) \sqsubseteq x_n \, .$$

has a smallest and a greatest solution.

Furthermore, these solutions will satisfy:

$$f_1(x_1, \ldots, x_n) = x_1, \ldots, f_n(x_1, \ldots, x_n) = x_n \, .$$

Proof: $\cdots$

Solving inequations over a lattice

Compute least solution for the lattice inequality system

$$f_1(x_1, \ldots, x_n) \sqsubseteq x_1, f_2(x_1, \ldots, x_n) \sqsubseteq x_2, \cdots, f_n(x_1, \ldots, x_n) \sqsubseteq x_n\,.$$

Initial solution: $x_1^0 = \bot, \ldots, x_n^0 = \bot$.

Iterative step:

$$\begin{aligned} x_1^{i+1} &= f_1(x_1^i, x_2^i, \ldots, x_n^i), \\ &\vdots \\ x_n^{i+1} &= f_n(x_1^i, x_2^i, \ldots, x_n^i)\,. \end{aligned}$$

Stopping criteria: $(\forall\ j \in [1, n])\ x_j^{i+1} \sqsubseteq x_j^i$.

Check that LHS functions are monotonic:

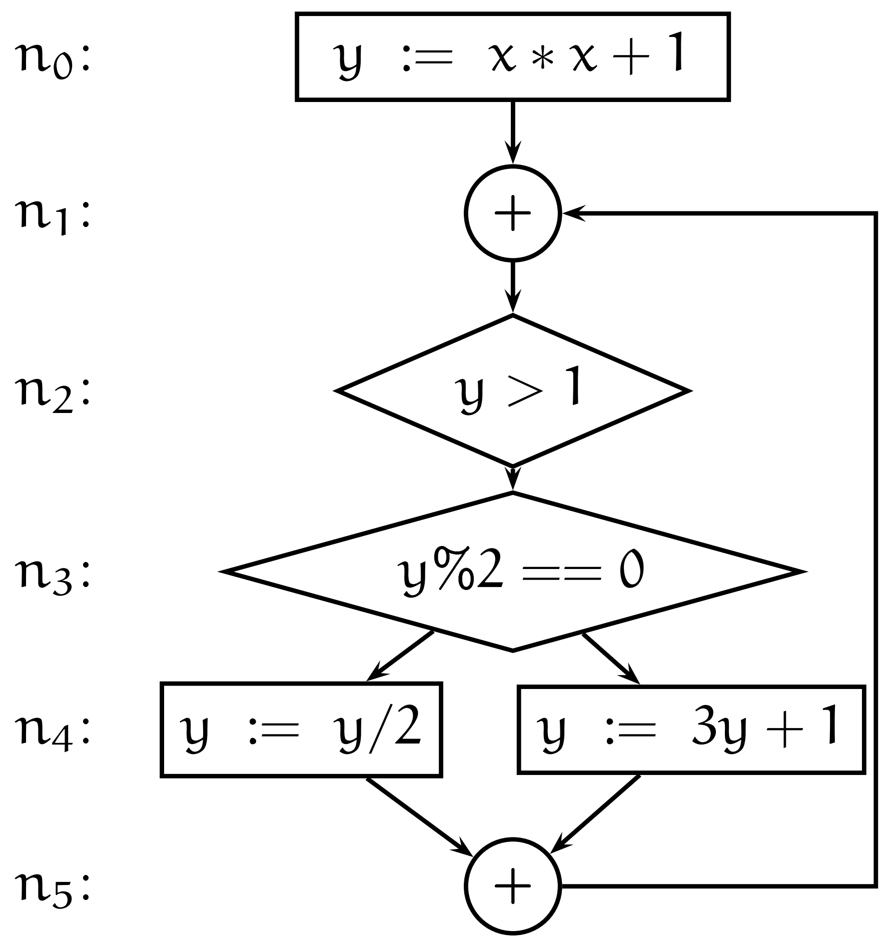
$$
\begin{array}{rcl}
\text{"}\top\text{"} & \sqsubseteq & \mathsf{sign}(n_0, y) \\
\mathsf{post}(n_0, \mathsf{sign}(n_0, x)) \sqcup \mathsf{sign}(n_5, y) & \sqsubseteq & \mathsf{sign}(n_1, y) \\
\mathsf{sign}(n_1, y) & \sqsubseteq & \mathsf{sign}(n_2, y) \\
\mathsf{sign}(n_2, y) \sqcap \text{"}+\text{"} & \sqsubseteq & \mathsf{sign}(n_3, y) \\
\mathsf{sign}(n_3, y) \sqcap \text{"}\top\text{"} & \sqsubseteq & \mathsf{sign}(n_4, y) \\
\mathsf{post}(n_4, \mathsf{sign}(n_4, y)) \sqcup \mathsf{post}(n_6, \mathsf{sign}(n_6, y)) & \sqsubseteq & \mathsf{sign}(n_5, y) \\
\hline
\text{"}+\text{"} & \sqsubseteq & \mathsf{sign}(n_0, x) \\
\mathsf{sign}(n_0, x) \sqcup \mathsf{sign}(n_5, x) & \sqsubseteq & \mathsf{sign}(n_1, x) \\
\mathsf{sign}(n_1, x) & \sqsubseteq & \mathsf{sign}(n_2, x) \\
 & \vdots & \\
\mathsf{sign}(n_4, x) \sqcup \mathsf{sign}(n_6, x) & \sqsubseteq & \mathsf{sign}(n5, x)
\end{array}
$$

Signs Analysis: Least Fixed Point Solution

x_0	y_0	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4	x_5	y_5	x_6	y_6
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$+$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$+$	$\top$	$+$	$+$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$+$	$\top$	$+$	$+$	$+$	$+$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$+$	$\top$	$+$	$+$	$+$	$+$	$+$	$+$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$+$	$\top$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$\bot$	$\bot$	$+$	$+$
$+$	$\top$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$\top$	$+$	$+$
$+$	$\top$	$+$	$\top$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$\top$	$+$	$+$
$+$	$\top$	$+$	$\top$	$+$	$\top$	$+$	$+$	$+$	$+$	$+$	$\top$	$+$	$+$
$+$	$\top$	$+$	$\top$	$+$	$\top$	$+$	$+$	$+$	$+$	$+$	$\top$	$+$	$+$

Signs Analysis: Final solution.

Note: $\mathsf{sign}(n, x) = $ "$+$" everywhere.



$\mathsf{sign}(n_0, y) : \top$

$\mathsf{sign}(n_1, y) : \top$

$\mathsf{sign}(n_2, y) : \top$

$\mathsf{sign}(n_3, y) : +$

$\mathsf{sign}(n_4(n_6), y) : +$

$\mathsf{sign}(n_5, y) : \top$

Solving Flow Inequations

Ascending Chain Condition:

There are no infinite chains: $a_1 \sqsubset a_2 \sqsubset \cdots \sqsubset \cdots$.

- If lattice L has ascending chain condition, then solution converges in $O(\mathrm{height}(L) * |CFG|)$.
- The lattice sign has height of 3.
- If height is not finite, then algorithm may not terminate.

Interval Analysis

Intervals: Basic Facts

Interval: $z \in [a, b] \stackrel{\Delta}{=} \{z | a \leq z \leq b\}$

We will consider intervals of integer values.

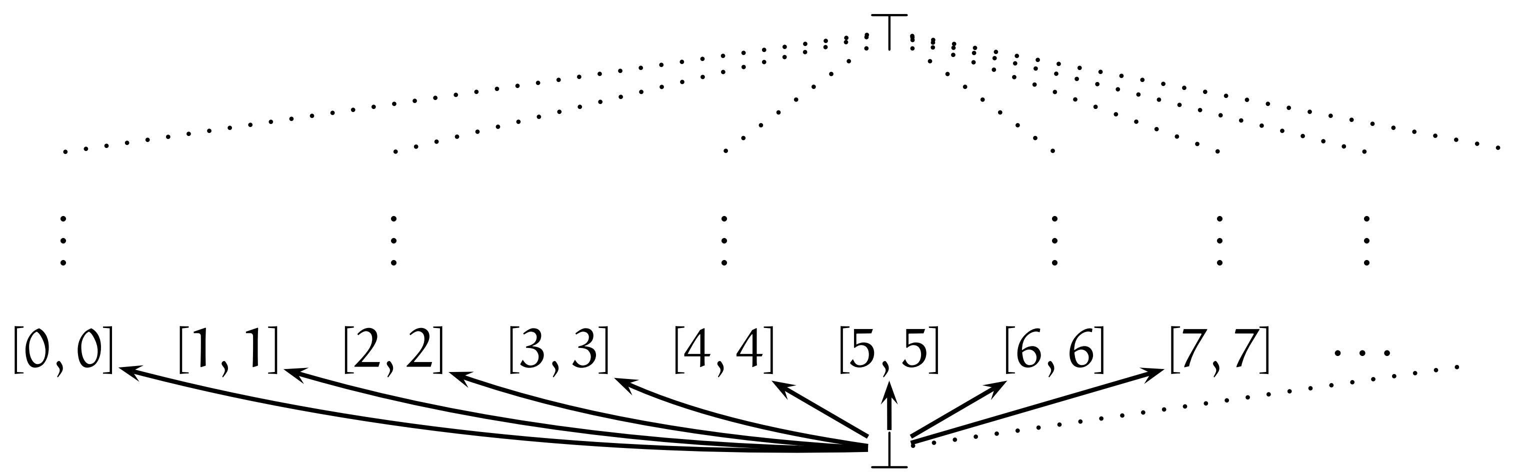
Half-open Intervals: $z \in [a, \infty)$ and $z \in (-\infty, a]$.

Interval Lattice: $[a, b] \sqsubseteq [c, d]$ iff $a \geq c \wedge b \leq d$.

Concretization: $[\![[a, b]]\!] = \{z | a \leq z \leq b\}$

Abstraction: Given $I \subseteq \mathsf{Z}$, $\alpha(I) = [\mathsf{min}_{\leq}(I), \mathsf{max}_{\leq}(I)]$.

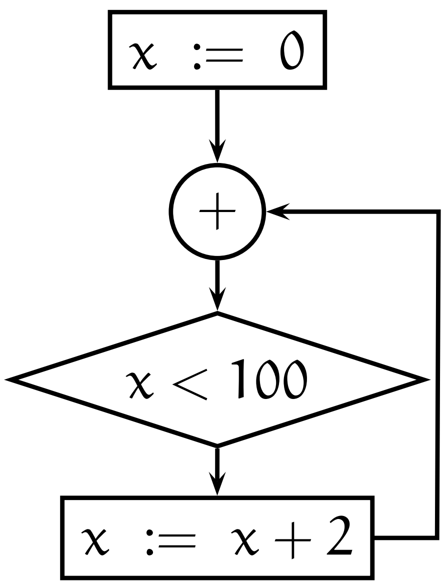
Interval Lattice



Note: Lattice is complete.

However, it has infinite height (and width).

Interval Analysis: Example # 2



n_0 : $(-\infty, \infty) \sqsubseteq \mathsf{Rng}(n_0, x)$

n_1 : $\mathsf{post}(x \ := \ 0, \mathsf{Rng}(n_0, x)) \sqsubseteq \mathsf{Rng}(n_1, x)$

n_2 : $\mathsf{Rng}(n_1, x) \sqsubseteq \mathsf{Rng}(n_2, x)$

n_3 : $\mathsf{Rng}(n_2, x) \sqcap \alpha([\![x < 100]\!]) \sqsubseteq \mathsf{Rng}(n_3, x)$

$\mathsf{post}(x \ := \ x + 2, \mathsf{Rng}(n_3, x)) \sqsubseteq \mathsf{Rng}(n_1, x)$

Interval Analysis: Solving Dataflow Equations

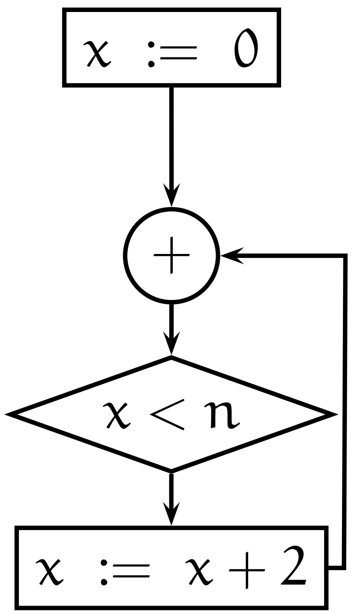
Notation: $R_i : \mathsf{Rng}(n_i, x)$.

$(-\infty, \infty) \sqsubseteq R_0$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\mathsf{post}(n_0, R_0) \sqcup \mathsf{post}(n_3, R_3) \sqsubseteq R_1$	$\bot$	$[0,0]$	$[0,0]$	$[0,0]$	$[0,1]$
$R_1 \sqsubseteq R_2$	$\bot$	$\bot$	$[0,0]$	$[0,0]$	$[0,0]$
$R_2 \sqcap [0,99] \sqsubseteq R_3$	$\bot$	$\bot$	$\bot$	$[0,0]$	$[0,0]$

This process converges in $100x$ steps to the following solution:

$$R_0 :\ \top,\ R_1 :\ [0,100],\ R_2 :\ [0,100],\ R_3 :\ [0,99]\,.$$

Interval Analysis: Example #3



n_0 : $(-\infty, \infty) \sqsubseteq \mathsf{Rng}(n_0, x)$

$[0, \infty) \sqsubseteq \mathsf{Rng}(n_0, n)$

n_1 : $\mathsf{post}(x \ := \ 0, \mathsf{Rng}(n_0, x)) \sqsubseteq \mathsf{Rng}(n_1, x)$

n_2 : $\mathsf{Rng}(n_1, x) \sqsubseteq \mathsf{Rng}(n_2, x)$

n_3 : $\mathsf{Rng}(n_2, x) \sqcap \alpha([\![x < n]\!]) \sqsubseteq \mathsf{Rng}(n_3, x)$

$\mathsf{post}(x \ := \ x + 2, \mathsf{Rng}(n_3, x)) \sqsubseteq \mathsf{Rng}(n_1, x)$

Solving Dataflow Equations

Notation: $R_i : \mathsf{Rng}(n_i, x)$.

$(-\infty, \infty) \sqsubseteq R_0$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\mathsf{post}(n_0, R_0) \sqcup \mathsf{post}(n_3, R_3) \sqsubseteq R_1$	$\bot$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 1]$
$R_1 \sqsubseteq R_2$	$\bot$	$\bot$	$[0, 0]$	$[0, 0]$	$[0, 0]$
$R_2 \sqcap [0, \infty) \sqsubseteq R_3$	$\bot$	$\bot$	$\bot$	$[0, 0]$	$[0, 0]$

This process does not converge in finitely many steps.

The least fixed point solution is:

$$R_0 : \top, \; R_1 : [0, \infty), \; R_2 : [0, \infty), \; R_3 : [0, \infty)\,.$$

Question: How do we compute fixed points in the interval lattice?

Widening

- The Interval lattice has infinite height.
- **Widening operator**: Let $[a, b] \sqsubseteq [c, d]$.

$$[a, b] \nabla [c, d] = [\ell, u]$$

wherein

$$\ell = \begin{cases} -\infty & \text{if } c < a, \\ c & \text{otherwise} \end{cases}$$

and similarly,

$$u = \begin{cases} \infty & \text{if } d > b, \\ b & \text{otherwise} \end{cases}$$

- Special case: $\bot \nabla i = i$.

Widening: Examples

Examples:

$$
\begin{array}{llllll}
[-1,-1] & \nabla & [-5,-5] & = & (-\infty,+\infty) \\
[1,1] & \nabla & [1,2] & = & [1,+\infty) \\
[1,1] & \nabla & [-1,1] & = & (-\infty,1] \\
[-1,5] & \nabla & [-1,5] & = & (-1,5) \\
\bot & \nabla & [10,100] & = & [10,100]
\end{array}
$$

Widening: Properties

Properties: The following properties are true of widening.

$$\text{(A)}\ (\forall x \sqsubseteq y)\ x \sqcup y \sqsubseteq x \nabla y$$

Let $a_1 \sqsubset a_2 \sqsubset a_3 \sqsubset \cdots$ be an increasing sequence.

Widened sequence: $b_1 = a_1$, $b_{i+1} = b_i \nabla (b_i \sqcup a_i)$.

Theorem: Widened sequence converges in finitely many steps, i.e.,

$$b_1 \sqsubseteq b_2 \sqsubseteq b_3 \cdots \sqsubseteq b_N = b_{N+1} = b_{N+2} \cdots .$$

and

$$\max_{\sqsubseteq}\{a_1, \ldots, \} \sqsubseteq b_N .$$

Widening: Application

Consider Dataflow Inequalities:

$$\begin{aligned} f_1(x_1, \ldots, x_n) &\sqsubseteq x_1 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &\sqsubseteq x_n \end{aligned}$$

Initial step: $\langle x_1^0, \ldots, x_n^0 \rangle = \langle \bot, \ldots, \bot \rangle$.

Widening Iteration: We split iteration into two steps:

Step 1: $\langle y_1^i, \ldots, y_n^i \rangle = \langle f_1(x_1^i, \ldots, x_n^i), \ldots, f_n(x_1^i, \ldots, x_n^i) \rangle$.

Step 2: $\langle x_1^{i+1}, \ldots, x_n^{i+1} \rangle = \langle x_1^i, \ldots, x_n^i \rangle \nabla \langle y_1^i, \ldots, y_n^i \rangle$.

Convergence: $(\forall j \in [1, n])\ y_j^N \sqsubseteq x_j^N$.

Widening: Example

Carry out the widening iteration for Example# 3.

Widening Iteration

- Widening iteration produces a solution to the dataflow inequalities.

$$(\forall j \in [1, n])\ f_j(x_1^N, \ldots, x_n^N) \sqsubseteq x_j\ . \qquad (1)$$

- However, there are two problems:

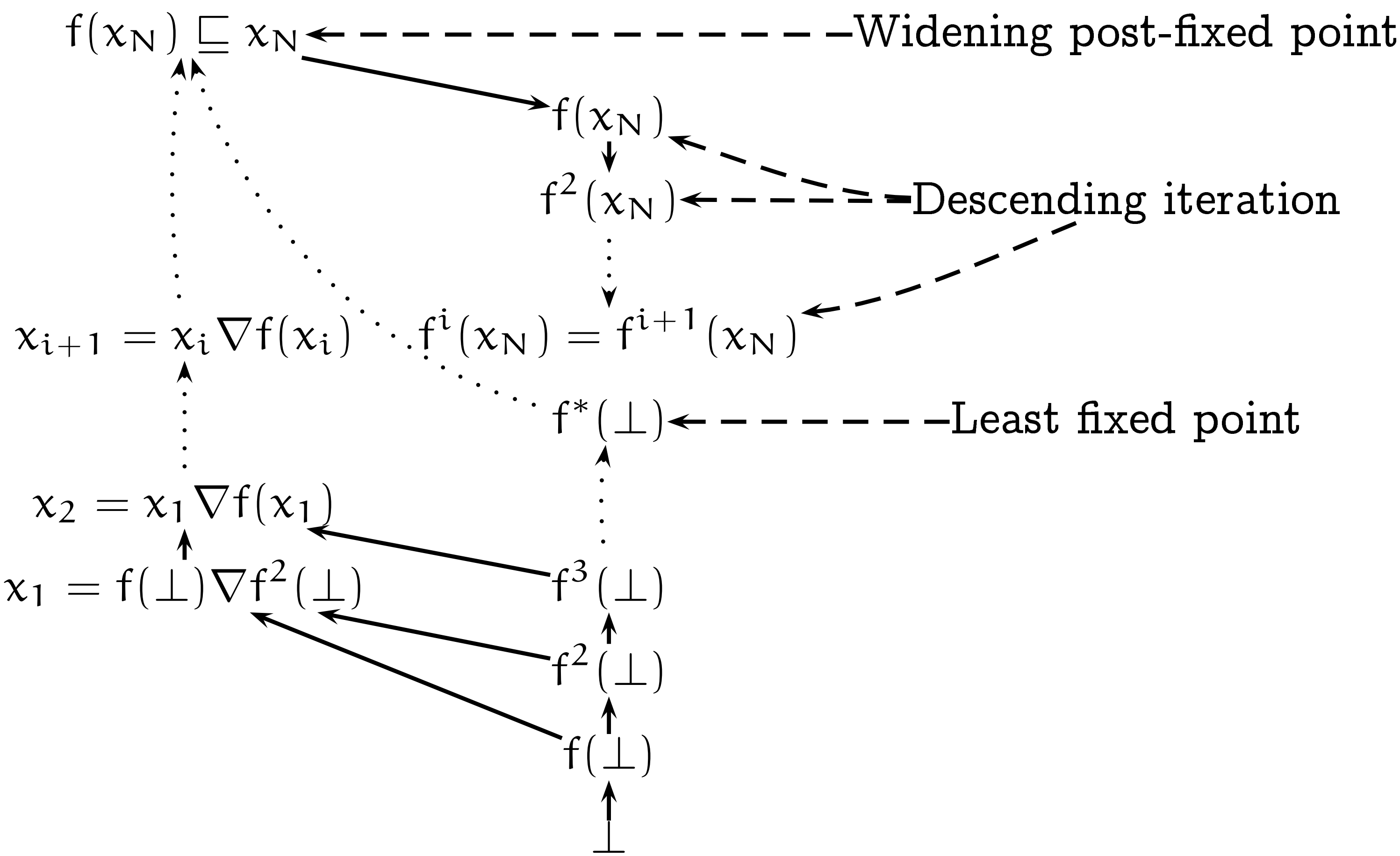
(a) Solution is no longer the least fixed point. Such solutions are called post-fixed points.

(b) Solution improvement may be possible. By monotonicity,

$$f(x) \sqsubseteq x \Rightarrow f(f(x)) \sqsubseteq f(x)\ .$$

Therefore, if x is a post-fixed point solution then $f(x)$ may be a smaller post-fixed point.

Ascending/Descending Iterations



Descending Iteration: Example #2

Carry out the widening iteration and descending iteration for Example# 2.

Descending Iteration: Convergence

Descending Chain Condition: Dual to Ascending Chain condition.

Descending iteration need not necessarily converge in finitely many steps.

(1) Stop the iteration after some fixed number of steps.
This is not a good idea (provide an example).

(2) Use a “narrowing” operator to force convergence.