Abstract interpretation, re-reloaded

Jan Midtgaard

Week 4, Abstract Interpretation

Aarhus University, Q4 - 2012

With figures courtesy of David A. Schmidt, Patrick and Radhia Cousot.

Last time

A first in-depth look at abstract interpretation based on Cousot-Cousot:JLP92.

- □ Foundations: Fixed points, Galois connections, . . .
- □ The Galois approach and friends: closure operators, Moore families, . . .
- □ From collecting semantics to analysis (soundness, optimality, completeness)
- □ The first step towards analysing Plotkin's three counter machine

Today

More approximation methods for abstract interpretation (Cousot-Cousot:JLP92):

- Partitioning
- Relational and attribute independent analysis
- Inducing, abstracting, approximating fixed points
- Widening, narrowing
- Forwards/backwards analysis

More fun with Plotkin's three counter machine

Relational vs. independent attribute analysis

Relational vs. independent attribute analysis

Definition. We say an analysis is *attribute independent*, if attributes are analysed independently of each other.

For example, the Parity analysis of `x`, `y`, and `z` we will develop, analyses the possible values of each variable in isolation.

Definition. We say an analysis is *relational*, if it can determine relations between attributes.

For example, imagine an analysis that can determine `x` is odd if and only if `y` is even.

Inducing, abstracting, and approximating fixed points

Fixed point inducing using Galois connections

Proposition. If $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$ is a Galois connection between posets $\langle C; \sqsubseteq, \sqcup \rangle$ and $\langle A; \leq, \vee \rangle$, $T : C \to C$ is such that $\operatorname{lfp} T = \bigsqcup_{n \geq 0} T^n(\bot)$, $\alpha(\bot_c) = \bot_a$, $T^\# : A \to A$ is such that $\alpha \circ T = T^\# \circ \alpha$ then $\alpha(\operatorname{lfp} T) = \bigvee_{n \geq 0} {T^\#}^n(\bot_a)$ and $\bigvee_{n \geq 0} {T^\#}^n(\bot_a)$ is a fixed point of $T^\#$. If $T^\# : A \to A$ is monotone, it is furthermore the *least* fixed point ($\geq \bot_a$).

Note: this proposition concerns a *complete approximation*.

Fixed point abstraction and approx. using Galois conn.

Proposition. If $\langle C; \sqsubseteq, \bot_c, \top_c, \sqcup, \sqcap\rangle$ and $\langle A; \leq, \bot_a, \top_a, \vee, \wedge\rangle$, are complete lattices, $\langle C; \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq\rangle$ and $F : C \to C$ is monotone, then $\alpha(\operatorname{lfp} F) \leq \operatorname{lfp}(\alpha \circ F \circ \gamma)$

Note: this proposition concerns an *optimal approximation* (akin to what you did for today).

Proposition. If $\langle A; \leq, \bot_a, \top_a, \vee, \wedge\rangle$ is a complete lattice, $T^{\#}, T^{\#'} : A \to A$ are monotone functions and $T^{\#} \mathrel{\dot{\leq}} T^{\#'}$, then $\operatorname{lfp} T^{\#} \leq \operatorname{lfp} T^{\#'}$.

Read: any monotone, upward judgement of the above composition will be fine.

Widening/narrowing reloaded

Narrowing motivation

We are after a (finite) approximation sequence $\check{X}^0 \geq \check{X}^1 \geq \check{X}^2 \geq \cdots \geq \check{X}^n \geq \operatorname{lfp} T^\#$ of the least fixed point (from above).

We could start from, e.g., $\check{X}^0 = \top$.

For the inductive step, not much is available: the previous iterate $\check{X}^k$ and the function $T^\#$. Assuming $\operatorname{lfp} T^\# \leq \check{X}^k$ and $T^\#$ is monotone, we want to ensure $\operatorname{lfp} T^\# \leq \check{X}^{k+1}$.

The narrowing operator $\vartriangle$ simply combines the available information:

$$\check{X}^{k+1} = \check{X}^k \vartriangle T^\#(\check{X}^k)$$

Narrowing definition

Definition. A narrowing operator $\vartriangle$ satisfies the following:

- For all $x, y : (x \vartriangle y) \leq x$ (ensure decr. seq.)
- For all $x, y, z : x \leq y \ \land \ x \leq z \implies x \leq (y \vartriangle z)$ (keep above)
- For any decreasing chain X_i the alternative chain defined as $\check{X}^0 = X_0$ and $\check{X}^{k+1} = \check{X}^k \vartriangle X_{k+1}$ stabilizes after a finite number of steps. (terminate)

Example: interval narrowing

Consider the domain of intervals: $\langle \wp(\mathbb{Z}); \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle Interval; \sqsubseteq \rangle$ defined as follows:

$$Interval = \{[l, u] \mid l \in \mathbb{Z} \cup \{-\infty\} \ \wedge \ u \in \mathbb{Z} \cup \{+\infty\} \ \wedge \ l \leq u\} \cup \emptyset$$

$$[a, b] \sqsubseteq [c, d] \iff c \leq a \ \wedge \ b \leq d$$

$$\alpha(\emptyset) = \emptyset$$

$$\alpha(X) = [\min X, \max X] \qquad \min \mathbb{Z} = -\infty \qquad \max \mathbb{Z} = +\infty$$

Strictly decreasing interval chains can be infinite:

$$[0, +\infty] \sqsupset [1, +\infty] \sqsupset [2, +\infty] \sqsupset \ldots$$

Hence we need a narrowing operator:

$$\emptyset \bigtriangleup I = \emptyset \qquad I \bigtriangleup \emptyset = \emptyset$$

$$[a, b] \bigtriangleup [c, d] = [\text{if } a = -\infty \text{ then } c \text{ else } a, \ \text{if } b = +\infty \text{ then } d \text{ else } b]$$

Downward iteration with narrowing

Proposition. If $T^\# : A \to A$ is a monotone function, and $\vartriangle : A \times A \to A$ is a narrowing operator and $T^\#(a) = a \leq a'$ then $\check{X}^0 = a', \ldots, \check{X}^{k+1} = \check{X}^k \vartriangle T^\#(\check{X}^k)$ converges with limit $\check{X}^n$, $n \in \mathbb{N}$ such that $a \leq \check{X}^n \leq a'$.

Intuition: this decreasing chain is finite and may take us closer to $T^\#$'s fixed point from above.

Note: In a complete lattice, if all strictly decreasing chains are finite, we can use $\vartriangle = \sqcap$.

Widening motivation

We aim for a better initial approximation than $\top$.

We are after a (finite) approximation sequence $\hat{X}^0 \leq \hat{X}^1 \leq \hat{X}^2 \leq \cdots \leq \hat{X}^n \geq \operatorname{lfp} T^{\#}$ of the least fixed point (starting below, ending above).

We could, e.g., try to iterate *above* a standard fixed point iteration: $X^0 = \bot, X^{k+1} = T^{\#}(X^k)$ towards $\operatorname{lfp} T^{\#}$.

Hence start from $\hat{X}^0 = \bot$

and use the widening operator $\triangledown$ to combine the available information:

$$\hat{X}^{k+1} = \hat{X}^k \triangledown T^{\#}(\hat{X}^k)$$

Widening definition

Definition. A widening operator satisfies the following:

- □ For all $x, y : x \leq (x \triangledown y) \ \wedge \ y \leq (x \triangledown y)$

  (keep above)

- □ For any increasing chain $X_0 \sqsubseteq X_1 \sqsubseteq X_2 \sqsubseteq \ldots$ the alternative chain defined as $\hat{X}^0 = X_0$ and $\hat{X}^{k+1} = \hat{X}^k \triangledown X_{k+1}$ stabilizes after a finite number of steps.

Example: interval widening

Consider again the domain of intervals:

$$\langle \wp(\mathbb{Z}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Interval; \sqsubseteq \rangle$$

For intervals strictly increasing chains can be infinite:

$$[0,0] \sqsubset [0,1] \sqsubset [0,2] \sqsubset \ldots$$

Hence we need a widening operator:

$$\emptyset \nabla I = I \qquad I \nabla \emptyset = I$$

$$[a,b] \nabla [c,d] = [\text{if } c < a \text{ then } -\infty \text{ else } a,\ \text{if } d > b \text{ then } +\infty \text{ else } b]$$

Upward iteration with widening

Proposition. If $T^\# : A \to A$ is a monotone function, and $\nabla : A \times A \to A$ is a widening operator then $\hat{X}^0 = \bot, \ldots, \hat{X}^{k+1} = \hat{X}^k \,\nabla\, T^\#(\hat{X}^k)$ converges with limit $\hat{X}^n$, $n \in \mathbb{N}$ such that $\operatorname{lfp} T^\# \leq \hat{X}^n$.

Note: In a complete lattice, if all strictly increasing chains are finite, we can use $\nabla = \sqcup$.

We don't actually need to widen in such a situation.

Forwards/backwards analysis

Transition systems with final states

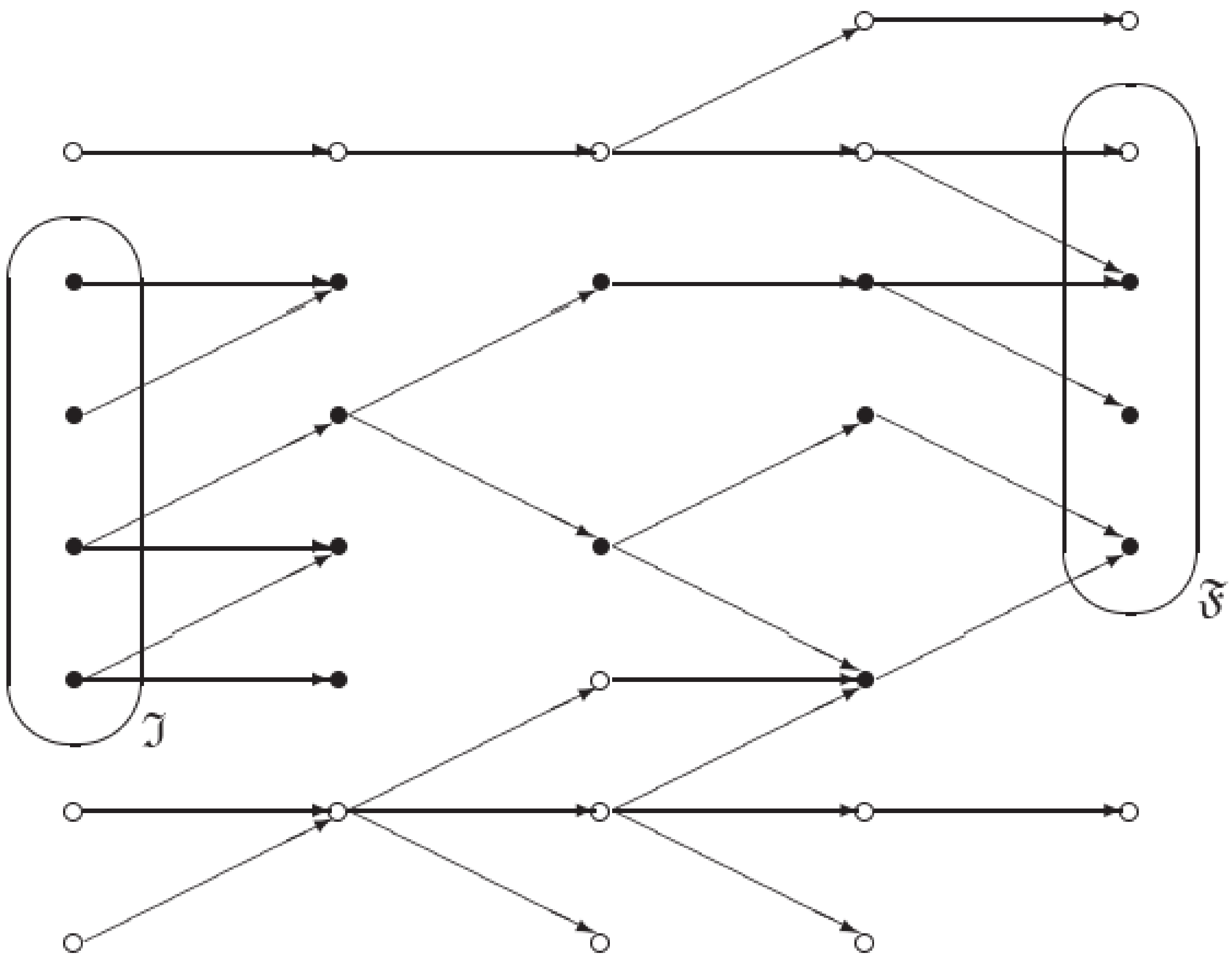
All though the transition system definition from week 1 included final states we haven't used them much:

Definition. A transition system is a quadruple $\langle S,\ I,\ F,\ \rightarrow\rangle$, where

- □ S is a set of states
- □ $I \subseteq S$ is a set of initial states
- □ $F \subseteq S$ is a set of final states ($\forall s \in F, s' \in S : s \not\rightarrow s'$)
- □ $\rightarrow \subseteq S \times S$ is a transition relation relating a state to its (possible) successors

Forwards collecting semantics (1/2)

Descendants of initial states (aka reachable states):



Forwards collecting semantics (2/2)

The forwards (top-down) collecting semantics can be expressed as a fixed point:

$$\begin{aligned}\operatorname{lfp} T \quad \text{where} \quad T(X) &= I \cup \{s \mid \exists s' \in X : s' \to s\} \\ &= I \cup \mathit{post}[\to](X)\end{aligned}$$

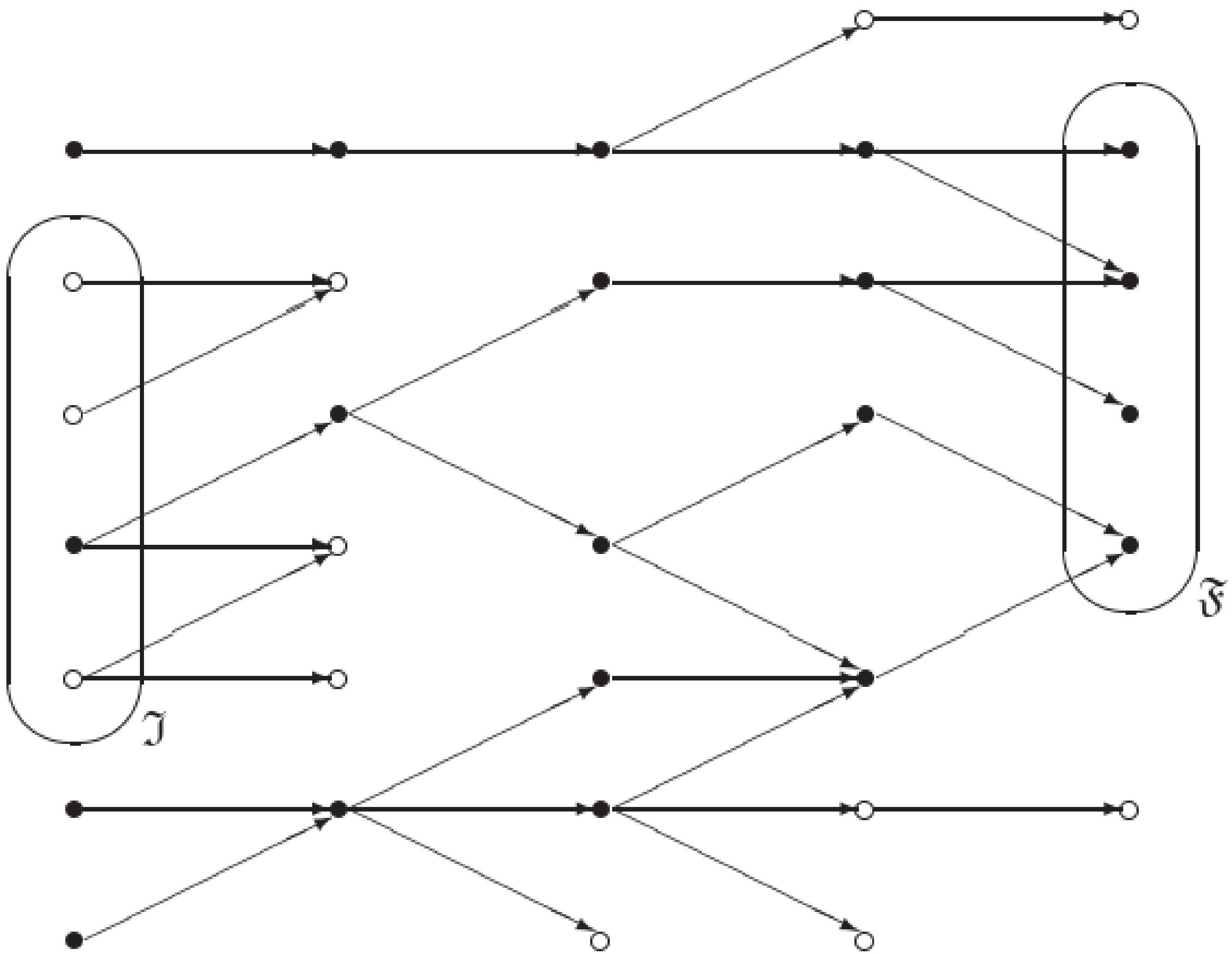
with

$$\mathit{post}[r](X) = \{s \mid \exists s' \in X : \langle s', s\rangle \in r\}$$

Note: here we are using the letter T for the transition function, as F is reserved for final states. . .

Backwards collecting semantics (1/2)

Ascendants of final states:



Backwards collecting semantics (2/2)

The backwards (bottom-up) collecting semantics can also be expressed as a fixed point:

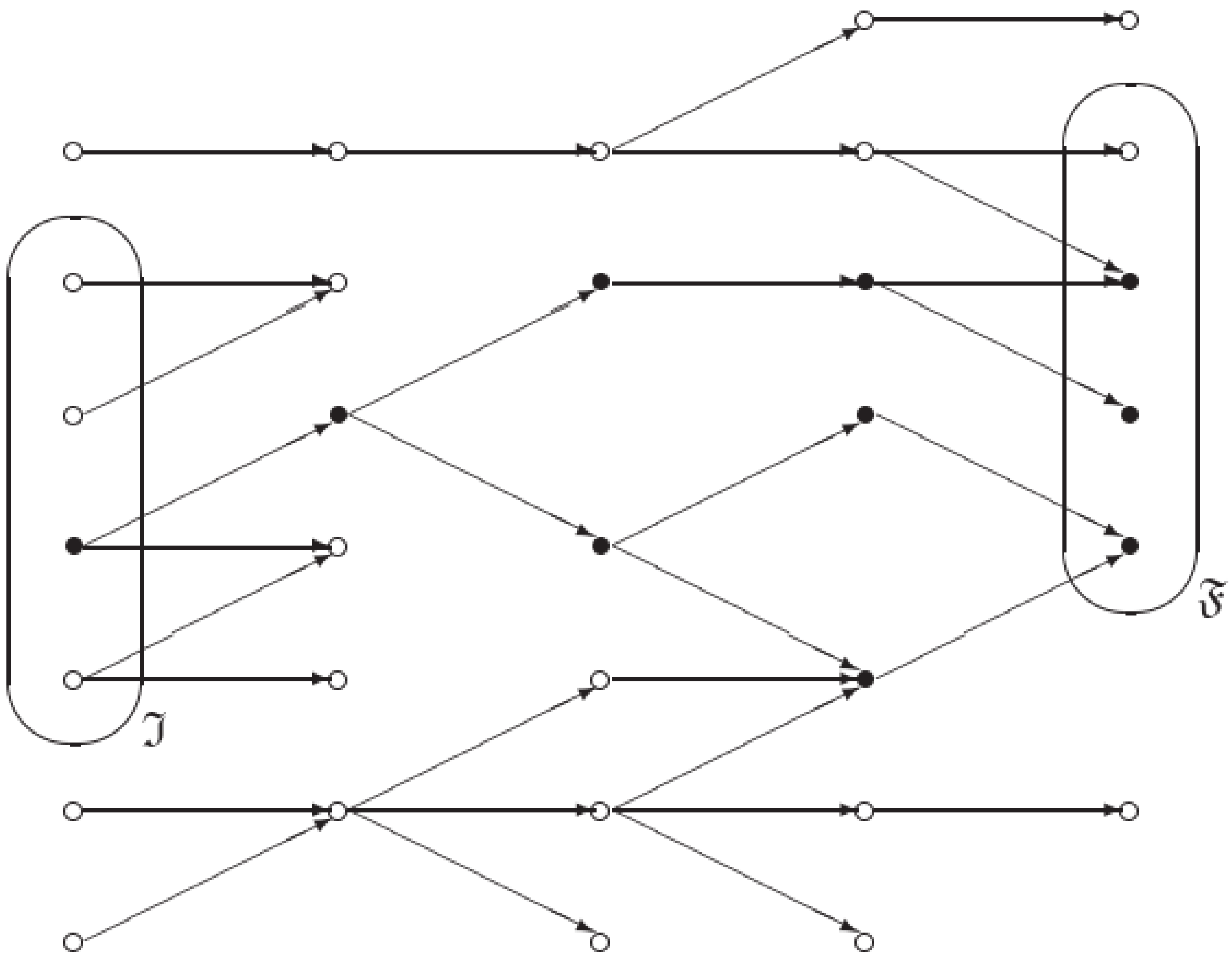
$$\begin{aligned}\operatorname{lfp} B \;\; \text{where} \;\; B(X) &= F \cup \{ s \mid \exists s' \in X : s \to s' \} \\ &= F \cup \mathit{pre}[\to](X)\end{aligned}$$

with

$$\mathit{pre}[r](X) = \{ s \mid \exists s' \in X : \langle s,\, s' \rangle \in r \}$$

Forwards/backwards collecting semantics (1/2)

Descendants of initial states which are also ascendants of final states:



Forwards/backwards collecting semantics (2/2)

This set of states can be expressed as the intersection of the two fixed points just defined:

$$\operatorname{lfp} T \cap \operatorname{lfp} B$$

The above is not computable in general, but the intuition is:

1. “run program forwards”,
2. “run program backwards”,
3. intersect.

Forwards/backwards collecting semantics in other words

We can express forwards/backward collecting semantics in several ways:

Proposition. Given a transition system $\langle S,\ I,\ F,\ \rightarrow\rangle$ with $X \subseteq S$, we have

1. $\mathit{pre}[\rightarrow](X) \cap \mathrm{lfp}\, T \subseteq \mathit{pre}[\rightarrow](X \cap \mathrm{lfp}\, T)$
2. $\mathit{post}[\rightarrow](X) \cap \mathrm{lfp}\, B \subseteq \mathit{post}[\rightarrow](X \cap \mathrm{lfp}\, B)$

   $\mathrm{lfp}\, T \cap \mathrm{lfp}\, B$

3. $= \mathrm{lfp}(\lambda X.\, \mathrm{lfp}\, T \cap B(X))$
4. $= \mathrm{lfp}(\lambda X.\, \mathrm{lfp}\, B \cap T(X))$
5. $= \mathrm{lfp}(\lambda X.\, \mathrm{lfp}\, T \cap \mathrm{lfp}\, B \cap B(X))$
6. $= \mathrm{lfp}(\lambda X.\, \mathrm{lfp}\, T \cap \mathrm{lfp}\, B \cap T(X))$

Forwards/backwards analysis

Once we move to an abstract domain, a sequence akin to the alternative characterizations is more precise:

Proposition. If $\langle C; \sqsubseteq, \bot_c, \top_c, \sqcup, \sqcap \rangle$ and $\langle A; \leq, \bot_a, \top_a, \vee, \wedge \rangle$, are complete lattices, $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$, $T, B : C \to C$ are monotone functions satisfying (5) and (6), $T^\#, B^\# : A \to A$ are monotone functions, such that $\alpha \circ T \circ \gamma \leq T^\#$ and $\alpha \circ B \circ \gamma \leq B^\#$, then the sequence

- $\dot{X}^0 = \operatorname{lfp} T^\#$ (or $\operatorname{lfp} B^\#$)
- $\dot{X}^{2n+1} = \operatorname{lfp}(\lambda X.\, \dot{X}^{2n} \wedge B^\#(X))$
- $\dot{X}^{2n+2} = \operatorname{lfp}(\lambda X.\, \dot{X}^{2n+1} \wedge T^\#(X))$

satisfies for all $k \in \mathbb{N} : \alpha(\operatorname{lfp} F \cap \operatorname{lfp} B) \leq \dot{X}^{k+1} \leq \dot{X}^k$

Hence, we have an ascending sequence.

Forwards/backwards analysis over infinite domains

We may also need to *narrow* in order to ensure termination of the downward iteration (if descending chains can be infinite):

$$\dot{X}^0 > \dot{X}^1 > \dot{X}^2 > \dots$$

Similarly, we may need to *widen* (and *narrow*) to ensure termination of the fixed point computation in each iterate.

- $\dot{X}^0 = \mathrm{lfp} \dots$
- $\dot{X}^{2n+1} = \mathrm{lfp}(\dots)$
- $\dot{X}^{2n+2} = \mathrm{lfp}(\dots)$

More fun with the three counter machine

Previously: analysing the 3 counter machine

```
  Var ::= x  |  y  |  z
 Inst ::= inc var  |  dec var  |  zero var m else n  |  stop
States  =  PC x \N_0 x \N_0 x \N_0
```

Transition relation:

```
<pc, xv, yv, zv> --> <pc+1, xv+1, yv, zv>                 if P_pc = inc x
        -        --> <pc+1, xv, yv+1, zv>                 if P_pc = inc y
        -        --> <pc+1, xv, yv, zv+1>                 if P_pc = inc z

<pc, xv, yv, zv> --> <pc+1, xv-1, yv, zv>         if P_pc = dec x /\ xv>0
        -        --> <pc+1, xv, yv-1, zv>         if P_pc = dec y /\ yv>0
        -        --> <pc+1, xv, yv, zv-1>         if P_pc = dec z /\ zv>0

<pc, xv, yv, zv> --> <pc', xv, yv, zv>      if P_pc = zero x pc' else pc''
                                               /\ xv=0
        -        --> <pc'', xv, yv, zv>     if P_pc = zero x pc' else pc''
                                               /\ xv<>0

<pc, xv, yv, zv> --> <pc', xv, yv, zv>      if P_pc = zero y pc' else pc''
                                               /\ yv=0
        -        --> <pc'', xv, yv, zv>     if P_pc = zero y pc' else pc''
                                               /\ yv<>0

<pc, xv, yv, zv> --> <pc', xv, yv, zv>      if P_pc = zero z pc' else pc''
                                               /\ zv=0
        -        --> <pc'', xv, yv, zv>     if P_pc = zero z pc' else pc''
                                               /\ zv<>0
```

We left off here:

```
T#(S#) =  Ø. [1 -> { <i,0,0> | i in N_0 }]

  U.
            U.         Ø. [pc+1 -> { <xv+1, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
      P_pc = inc x                (...and for y and z)
  U.
            U.         Ø. [pc+1 -> { <xv-1, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
      P_pc = dec x
          xv>0                    (...and for y and z)
  U.
            U.         Ø. [pc' -> { <xv, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
  P_pc = zero x pc' else pc''
           xv=0                   (...and for y and z)
  U.
            U.         Ø. [pc'' -> { <xv, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
  P_pc = zero x pc' else pc''
          xv<>0                   (...and for y and z)
```

Call-by-need Galois connections :-) (1/3)

Abstracting a set valued function:

Given a Galois connection between complete lattices, we can lift it pointwise to function spaces (also complete lattices):

$$\frac{\langle \wp(C); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \sqsubseteq \rangle}{\langle D \to \wp(C); \dot{\subseteq} \rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle D \to A; \dot{\sqsubseteq} \rangle}$$

where $$\begin{aligned} \dot{\alpha}(F) &= \lambda d.\, \alpha(F(d)) \\ \dot{\gamma}(F^{\#}) &= \lambda d.\, \gamma(F^{\#}(d)) \end{aligned}$$

Call-by-need Galois connections :-) (2/3)

Abstracting a set of triples by a triple of sets:

$$\overline{\langle \wp(A \times B \times C); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(A) \times \wp(B) \times \wp(C); \subseteq_\times \rangle}$$

between complete lattices (the latter being reduced) where

$$\begin{aligned} \subseteq_\times &= \ \subseteq \times \subseteq \times \subseteq \\ \alpha(T) &= \langle \pi_1(T),\ \pi_2(T),\ \pi_3(T) \rangle \\ \gamma(\langle X,\ Y,\ Z \rangle) &= X \times Y \times Z \end{aligned}$$

Call-by-need Galois connections :-) (3/3)

Abstracting a triple of sets by an abstract triple:

Given three Galois connections between complete lattices, we can form a new Galois connection (also over complete lattices):

$$\dfrac{\langle \wp(A); \subseteq \rangle \underset{\alpha_A}{\overset{\gamma_A}{\leftrightarrows}} \langle A'; \sqsubseteq_a \rangle \qquad \langle \wp(B); \subseteq \rangle \underset{\alpha_B}{\overset{\gamma_B}{\leftrightarrows}} \langle B'; \sqsubseteq_b \rangle \qquad \langle \wp(C); \subseteq \rangle \underset{\alpha_C}{\overset{\gamma_C}{\leftrightarrows}} \langle C'; \sqsubseteq_c \rangle}{\langle \wp(A) \times \wp(B) \times \wp(C); \subseteq_\times \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A' \times B' \times C'; \sqsubseteq_\times \rangle}$$

where

$$\begin{aligned} \subseteq_\times &= \subseteq \times \subseteq \times \subseteq \\ \sqsubseteq_\times &= \sqsubseteq_a \times \sqsubseteq_b \times \sqsubseteq_c \\ \alpha(\langle X, Y, Z \rangle) &= \langle \alpha_A(X), \alpha_B(Y), \alpha_C(Z) \rangle \\ \gamma(\langle X', Y', Z' \rangle) &= \langle \gamma_A(X), \gamma_B(Y), \gamma_C(Z) \rangle \end{aligned}$$

Three counter analysis from 10000 feet[1]

The Parity analysis is composed in two.

Last week:

$$\dfrac{}{\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows PC \rightarrow \wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0)}$$

This week:

$$\dfrac{\dfrac{\dfrac{}{\wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows \wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0)} \quad \dfrac{\dfrac{}{\wp(\mathbb{N}_0) \underset{\twoheadrightarrow}{\longleftarrow} Par} \quad \dfrac{}{\wp(\mathbb{N}_0) \underset{\twoheadrightarrow}{\longleftarrow} Par} \quad \dfrac{}{\wp(\mathbb{N}_0) \underset{\twoheadrightarrow}{\longleftarrow} Par}}{\wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0) \leftrightarrows Par \times Par \times Par}}{\wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows Par \times Par \times Par}}{PC \rightarrow \wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows PC \rightarrow Par \times Par \times Par}$$

Hence by transitivity:

$$\dfrac{}{\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows PC \rightarrow Par \times Par \times Par}$$

1. and therefore in a very small font

At home: operators/property transformers

At home you calculated abstract operators:

```
 =0 : Parity -> Parity
<>0 : Parity -> Parity
 +1 : Parity -> Parity
 -1 : Parity -> Parity
```

from concrete ones over $\mathbb{N}_0$:

```
 =0 : \N_0 -> \N_0
    = \S. {s | s in S /\ s=0 }
<>0 : \N_0 -> \N_0
    = \S. {s | s in S /\ s<>0 }
 +1 : \N_0 -> \N_0
    = \S. {s+1 | s in S}
 -1 : \N_0 -> \N_0
    = \S. {s-1 | s in S /\ s>0 }
```

Result

```
T(S#) =  ( <bot,bot,bot>. [1 -> <top, even, even> ] )

  U.
            U.          ( <bot,bot,bot>. [pc+1 -> [x++]#(S#(pc))] )
      pc in Dom(S#)
      P_pc = inc x

            (...and for y and z)

  U.
            U.          ( <bot,bot,bot>. [pc+1 -> [x--]#(S#(pc))] )
      pc in Dom(S#)
      P_pc = dec x

            (...and for y and z)

  U.
            U.          ( <bot,bot,bot>. [pc' -> [x==0]#(S#(pc))] )
      pc in Dom(S#)            U. ( <bot,bot,bot>. [pc'' -> [x<>0]#(S#(pc))] )
  P_pc = zero x pc' else pc''

            (...and for y and z)
```

Summary

Summary

More approximation methods for abstract interpretation (Cousot-Cousot:JLP92):

- □ Partitioning
- □ Relational and attribute independent analysis
- □ Inducing, abstracting, approximating fixed points
- □ Widening, narrowing
- □ Forwards/backwards analysis

+ analysis of Plotkin’s three counter machine