Abstract Induction

ETH Zürich
October 2–3, 2015

Patrick Cousot
pcousot@cs.nyu.edu cs.nyu.edu/~pcousot

Concrete Induction

Software correctness proofs

- Any formal proof of a non-trivial program requires a reasoning by mathematical induction (e.g., following Turing, on the number of program execution steps):
- Invent an inductive argument (e.g. invariant, variant function), the hardest part
- Prove the base case and inductive case (e.g. true on loop entry and preserved by one more loop iteration)
- Prove that the inductive argument is strong-enough, that is, it implies the program property to be verified

Avoiding the difficulties: (1) finitary methods

Avoiding the difficulty

- Unsoundness: not for scientists
- Model-checking: finite enumeration, no induction needed
- Deductive methods (theorem provers, proof verifiers, SMT solvers): avoid (part of) the difficulty since the inductive argument must be provided by the end-user ($\Longrightarrow$ still difficult, shame is on the prover)
- Finitary abstractions (predicate abstraction $\equiv$ any finite abstract domain): only finitely many possible statements to be checked to be inductive

Limitations of finite abstractions

- A sound and complete finite abstraction exists to prove any property of any program:

`x=0; while x<1 do x++` $\longrightarrow \{\bot, [0,0], [0,1], [-\infty,\infty]\}$

`x=0; while x<2 do x++` $\longrightarrow \{\bot, [0,0], [0,1], [0,2], [-\infty,\infty]\}$

…

`x=0; while x<`n `do x++` $\longrightarrow \{\bot, [0,0], [0,1], [0,2], [0,3], \ldots, [0,n], [-\infty,\infty]\}$

…

- Not true for a programming language !
- Finite abstractions fail on infinitely many programs on which infinitary abstractions do succeed

Avoiding the difficulty (II) Refinement in finite domains

Verification/static analysis by abstract interpretation

- Define the abstraction:

$$\langle \wp(\mathscr{D}[\![\mathtt{P}]\!]), \subseteq \rangle \underset{\alpha[\![\mathtt{P}]\!]}{\overset{\gamma[\![\mathtt{P}]\!]}{\leftrightarrows}} \langle \mathscr{A}[\![\mathtt{P}]\!], \sqsubseteq \rangle$$

- Calculate the abstract semantics:

$$S^{\#}[\![\mathtt{P}]\!] = \alpha[\![\mathtt{P}]\!](\{S[\![\mathtt{P}]\!]\})$$ exact abstraction

$$S^{\#}[\![\mathtt{P}]\!] \sqsupseteq \alpha[\![\mathtt{P}]\!](\{S[\![\mathtt{P}]\!]\})$$ approximate abstraction

- Soundness (by construction):

$$\forall\, \mathtt{P} \in \mathbb{L}: \forall\, Q \in \mathscr{A}: S^{\#}[\![\mathtt{P}]\!] \sqsubseteq Q \implies S[\![\mathtt{P}]\!] \in \gamma[\![\mathtt{P}]\!](Q)$$

Refinement: good news

- Problem: how to prove a valid abstract property $\alpha(\{\mathsf{lfp}\ F[\![\mathtt{P}]\!]\}) \sqsubseteq Q$ when $\alpha \circ F \sqsubseteq F^{\#} \circ \alpha$ but $\mathsf{lfp}\ F^{\#}[\![\mathtt{P}]\!] \not\sqsubseteq Q$? (i.e. strongest inductive argument too weak)

- It is always possible to refine $\langle \mathscr{A}, \sqsubseteq \rangle$ into a most abstract more precise abstraction $\langle \mathscr{A}', \sqsubseteq' \rangle$ such that

$$\langle \wp(\mathscr{D}), \subseteq \rangle \underset{\alpha'}{\overset{\gamma'}{\leftrightarrows}} \langle \mathscr{A}', \sqsubseteq' \rangle$$

and $\alpha' \circ F = F' \circ \alpha$ with $\mathsf{lfp}\ F'[\![\mathtt{P}]\!] \sqsubseteq' \alpha' \circ \gamma\ (Q)$

(thus proving $\mathsf{lfp}\ F[\![\mathtt{P}]\!] \in \gamma'(Q)$ which implies $\mathsf{lfp}\ F[\![\mathtt{P}]\!] \in \gamma(Q)$)

Roberto Giacobazzi, Francesco Ranzato, Francesca Scozzari: Making abstract interpretations complete. J. ACM 47(2): 361-416 (2000)

Refinement: bad news

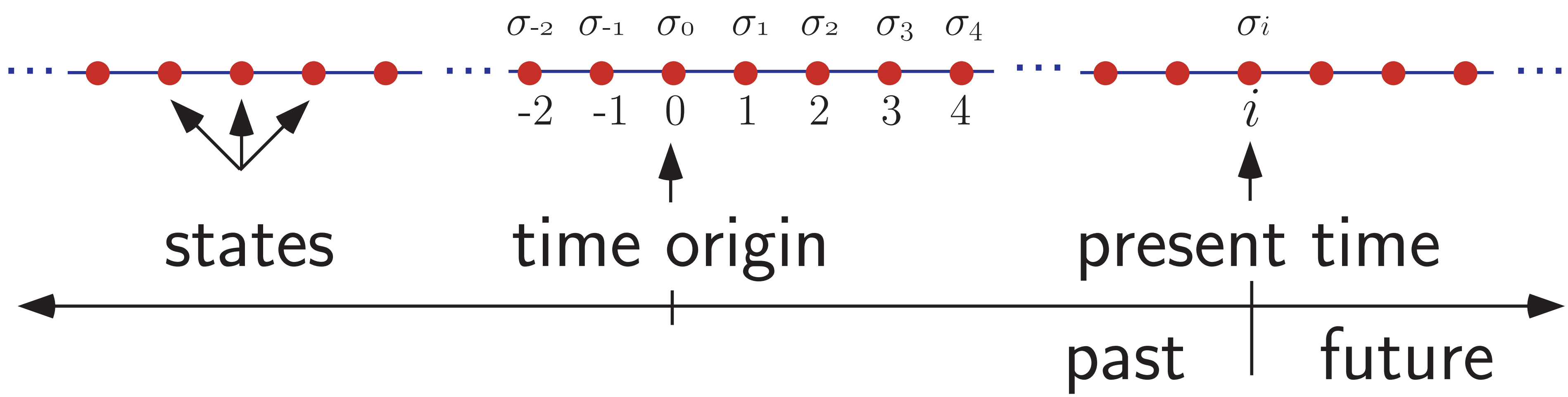
- But, refinements of an abstraction can be intrinsically incomplete
- The only complete refinement of that abstraction for the collecting semantics is :

  the identity (i.e. no abstraction at all)

- In that case, the only complete refinement of the abstraction is to the collecting semantics and any other refinement is always imprecise

Example of intrinsic approximate refinement

- Consider executions traces $\langle i, \sigma \rangle$ with infinite past and future:



Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

Example of intrinsic approximate refinement

- Consider the temporal specification language $\overset{\curvearrowleft}{\mu}{}^{\star}$ (containing LTL, CTL, CTL*, and Kozen's μ-calculus as fragments):

$$
\begin{array}{rcll}
\varphi & ::= & \boldsymbol{\sigma}_S & S \in \wp(\mathbb{S}) \qquad \text{state predicate} \\
& \mid & \boldsymbol{\pi}_t & t \in \wp(\mathbb{S} \times \mathbb{S}) \qquad \text{transition predicate} \\
& \mid & \oplus\, \varphi_1 & \qquad \text{next} \\
& \mid & \varphi_1^{\curvearrowleft} & \qquad \text{reversal} \\
& \mid & \varphi_1 \vee \varphi_2 & \qquad \text{disjunction} \\
& \mid & \neg\, \varphi_1 & \qquad \text{negation} \\
& \mid & X & X \in \mathbb{X} \qquad \text{variable} \\
& \mid & \boldsymbol{\mu}\, X \cdot \varphi_1 & \qquad \text{least fixpoint} \\
& \mid & \boldsymbol{\nu}\, X \cdot \varphi_1 & \qquad \text{greatest fixpoint} \\
& \mid & \forall\, \varphi_1 : \varphi_2 & \qquad \text{universal state closure}
\end{array}
$$

Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

Example of intrinsic approximate refinement

- Consider universal model-checking abstraction:

$$\mathrm{MC}^{\forall}_{M}(\phi) = \alpha^{\forall}_{M}(\llbracket\phi\rrbracket) \in \wp(\mathit{Traces}) \rightarrow \wp(\mathit{States})$$

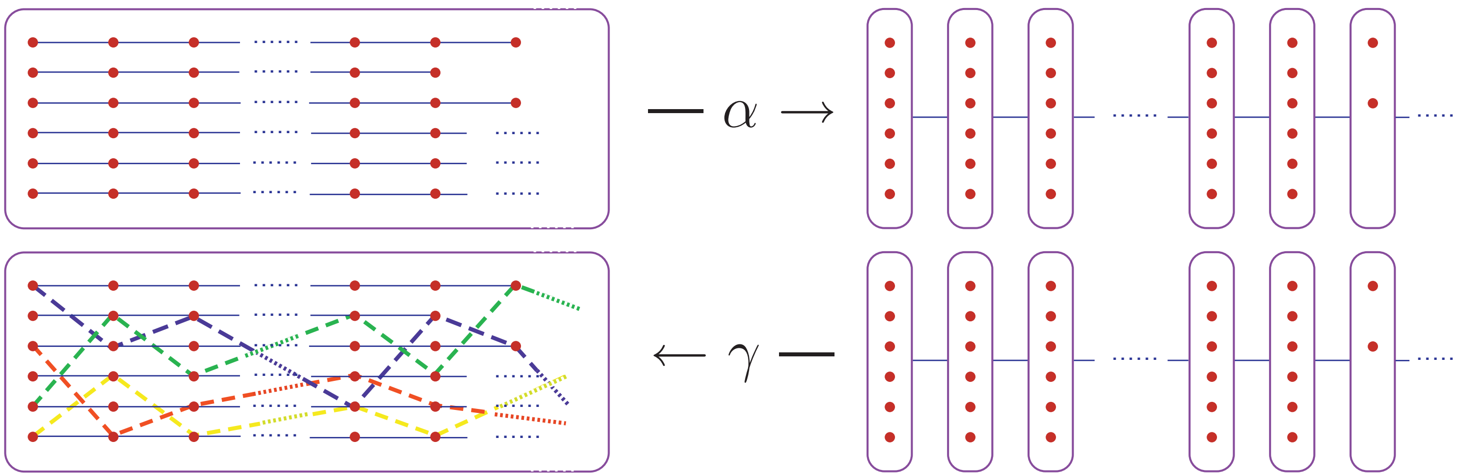
$$= \{s \in \mathit{States} \mid \forall\langle i,\sigma\rangle \in \mathit{Traces}_M \,.\, (\sigma_i = s) \Rightarrow \langle i,\sigma\rangle \in \llbracket\phi\rrbracket\}$$

where M is defined by a transition system

(and dually the existential model-checking abstraction)

Example of intrinsic approximate refinement

- The abstraction from a set of traces to a trace of sets is sound but *incomplete*, even for finite systems (*)

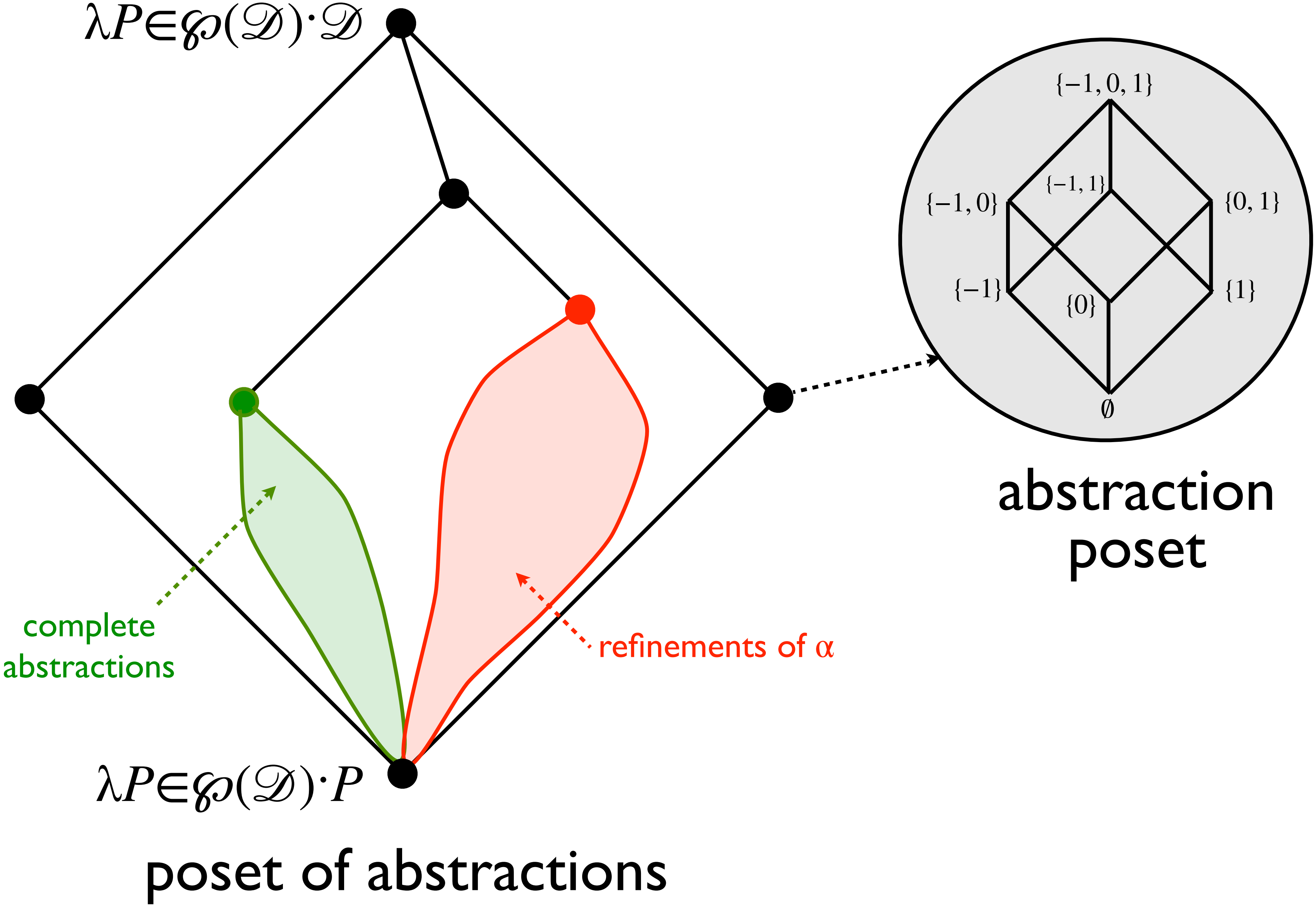


- Any refinement of this abstraction is *incomplete* (but to the infinite past/future trace semantics itself) (**)

(*) Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

(**) Roberto Giacobazzi, Francesco Ranzato: Incompleteness of states w.r.t. traces in model checking. Inf. Comput. 204(3): 376-407 (2006)

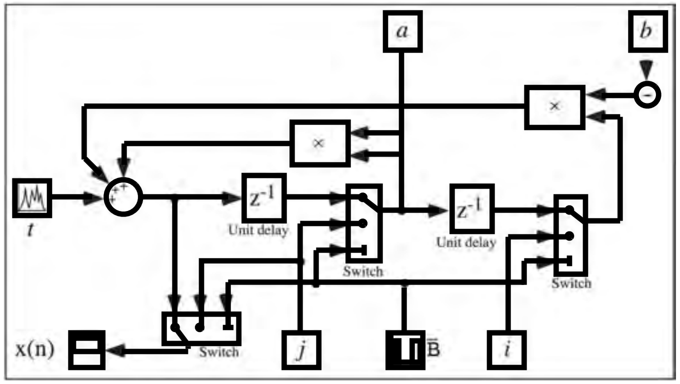
Intrinsic approximate refinement



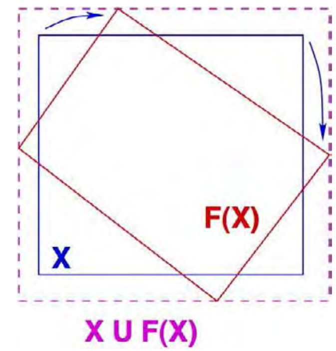
In general refinement does not terminate

- Example:filter invariant abstraction:

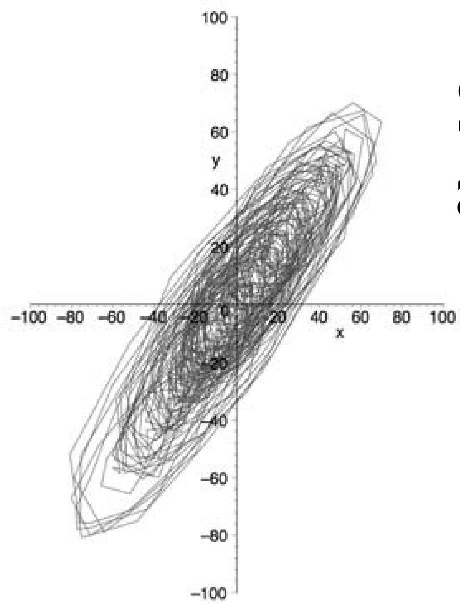
2nd order filter:



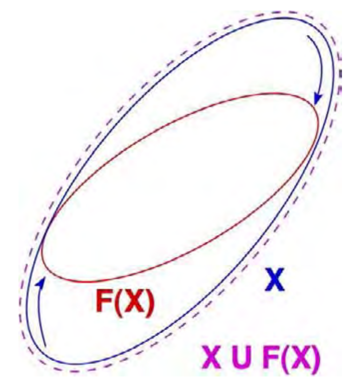
Unstable polyhedral abstraction:



Counter-example guided refinement will indefinitely add missing points according to the execution trace:



Stable ellipsoidal abstraction:



Julien Bertrane, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, & Xavier Rival. Static Analysis and Verification of Aerospace Software by Abstract Interpretation. In *AIAA Infotech@@Aerospace 2010*, Atlanta, Georgia. American Institute of Aeronautics and Astronautics, 20—22 April 2010. © AIAA.

In general refinement does not terminate

- Narrowing is needed to stop infinite iterated automatic refinements:

  e.g. SLAM stops refinement after 20mn, now abandoned (despite complete success claimed in 98% of studied cases (*))

- Intelligence is needed for refinement:

  e.g. human-driven refinement of Astrée (**)

(*) Thomas Ball, Vladimir Levin, Sriram K. Rajamani: A decade of software model checking with SLAM. Commun. ACM 54(7): 68-76 (2011)

(**) Julien Bertrane, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, & Xavier Rival. Static Analysis and Verification of Aerospace Software by Abstract Interpretation. In *AIAA Infotech@@Aerospace 2010*, Atlanta, Georgia. American Institute of Aeronautics and Astronautics, 20—22 April 2010. © AIAA.

Facing the difficulties: Abstract induction

Sound software static analysis

- The mathematical induction must be performed in the abstract (e.g. the inductive argument must belong to an abstract domain with a finite computer representation)
- (and imply the mathematical induction in the concrete)

Abstract induction

- The inductive argument must be expressible in the abstract domain (complex abstract domains favored)
- It must be strong enough to imply the program property (complex abstract domains favored
- It must be inferable in the abstract (simple abstract domains favored)

Abstract induction in infinite domains

Abstract Interpreters

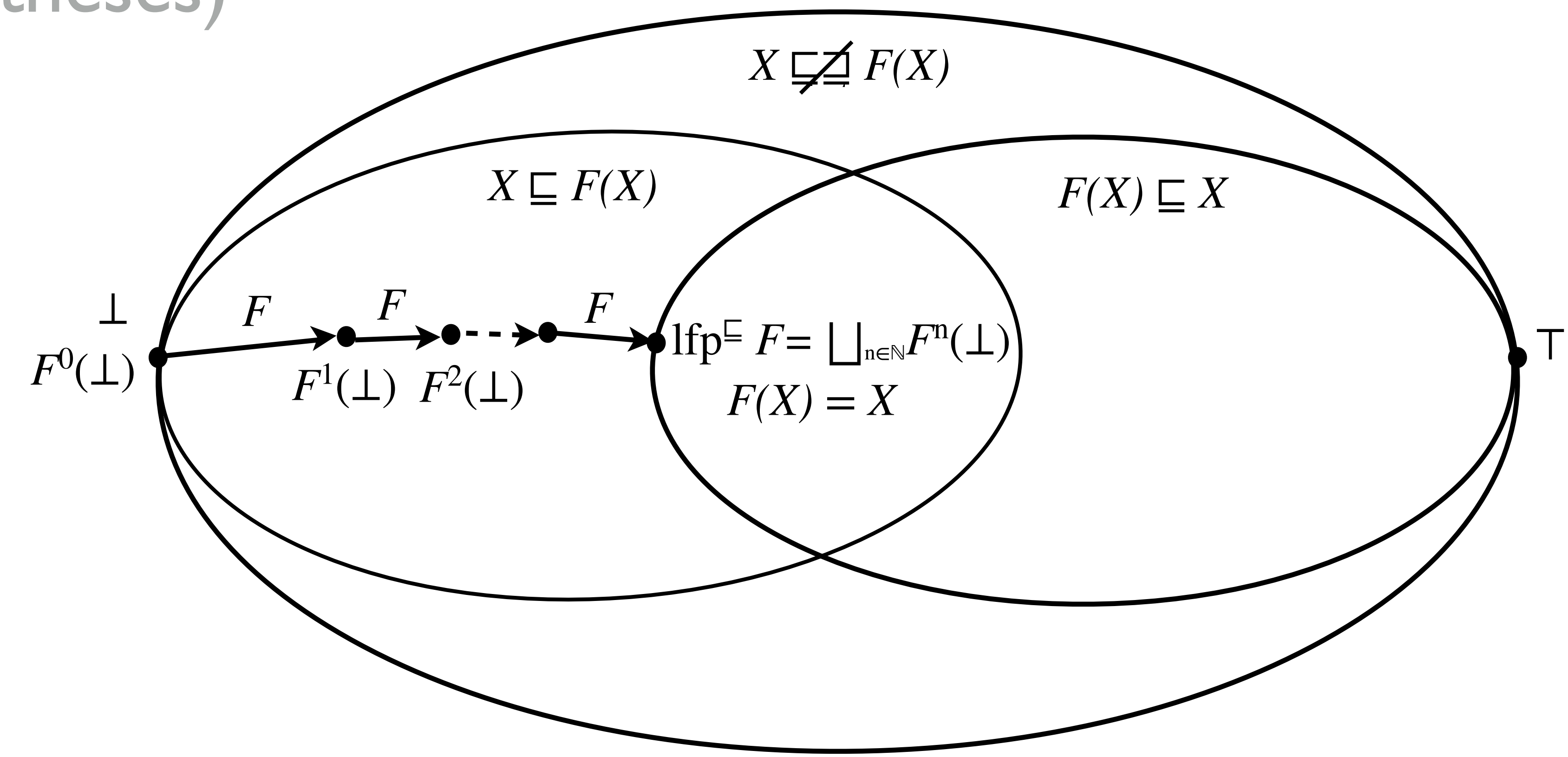
- Transitional abstract interpreters: proceed by induction on program steps
- Structural abstract interpreters: proceed by induction on the program syntax
- Common main problem: over/under-approximate fixpoints in non-Noetherian(*) abstract domains (**)

(*) Iterative fixpoint computations may not converge in finitely many steps

(**) Or convergence may be guaranteed but to slow.

Fixpoints

- Poset (or pre-order) $\langle D, \sqsubseteq, \perp, \sqcup \rangle$
- Transformer (increasing in the concrete) $F \in D \longmapsto D$
- Least fixpoint: $\mathrm{lfp}^{\sqsubseteq} F = \bigsqcup_{n\in\mathbb{N}} F^n(\perp)$ (under appropriate hypotheses)



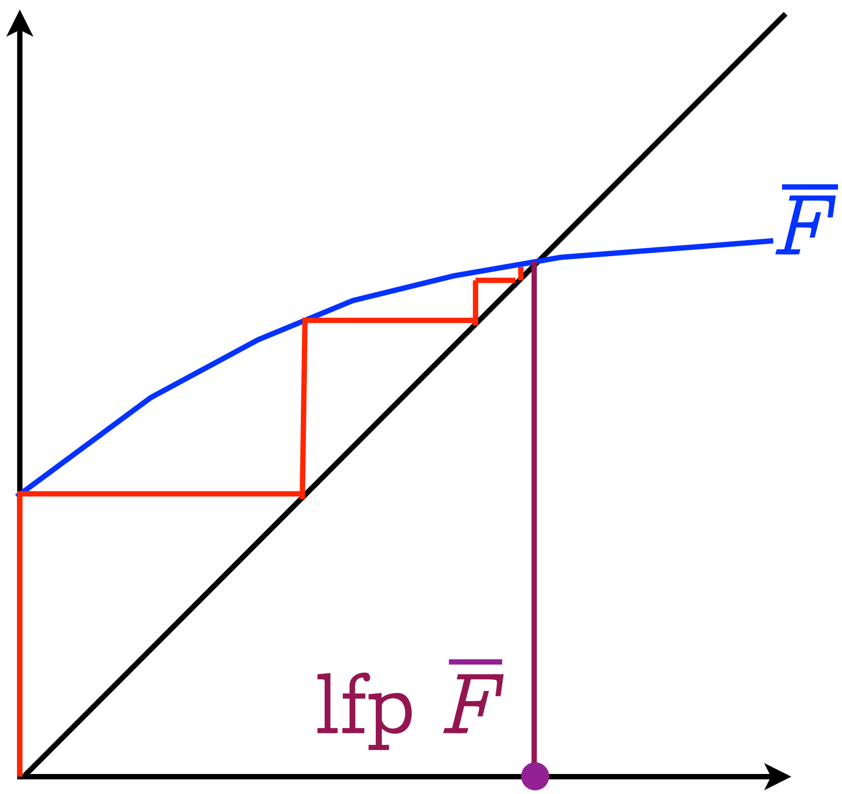
Convergence criterion

- By Tarski (or variants)

$$F(X) \sqsubseteq X \quad \Longrightarrow \quad \mathrm{lfp}^{\sqsubseteq} F \sqsubseteq X$$

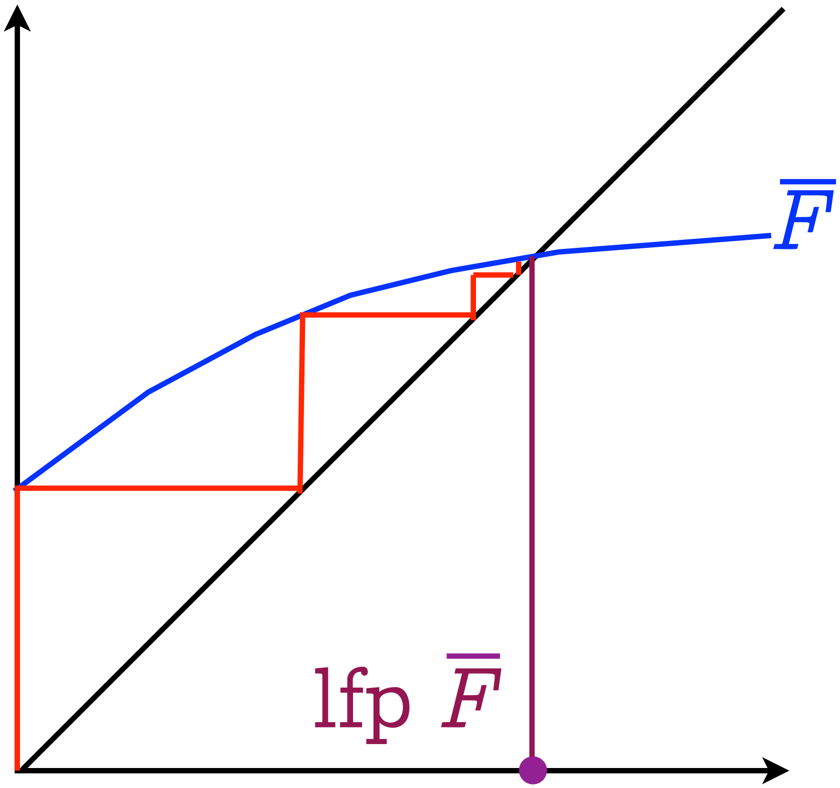
Widening

Convergence acceleration with widening

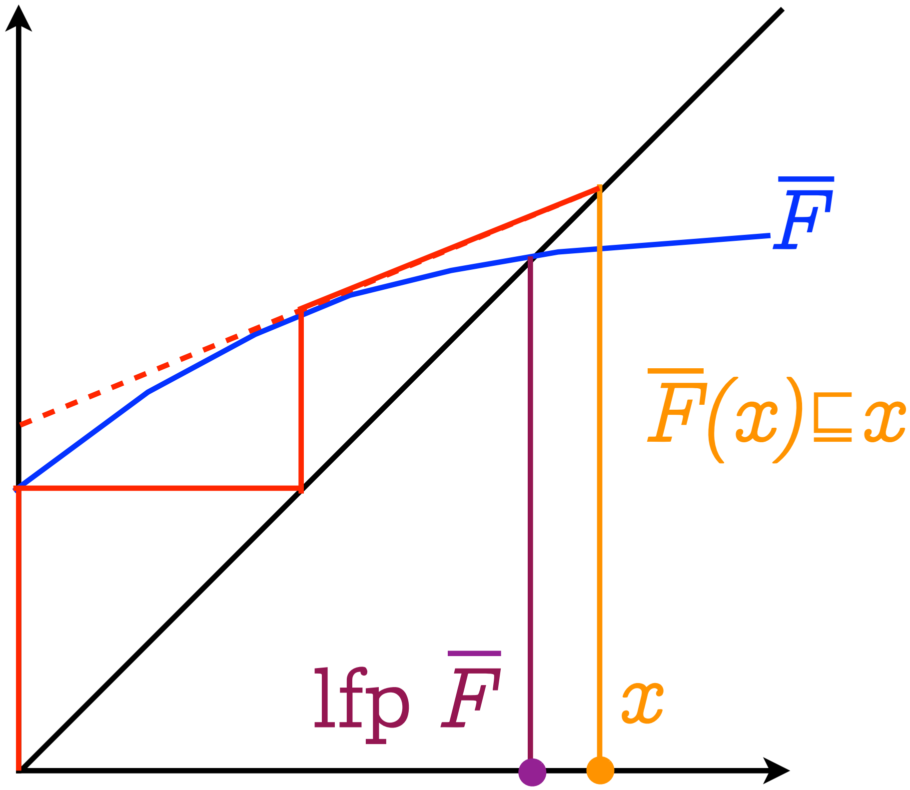


Infinite iteration

Convergence acceleration with widening



Infinite iteration



Accelerated iteration with widening
(e.g. with a widening based on the derivative as in Newton-Raphson method(*))

(*) Javier Esparza, Stefan Kiefer, Michael Luttenberger: Newtonian program analysis. J. ACM 57(6): 33 (2010)

Extrapolation by Widening

- $X^0 = \perp$ (increasing iterates with widening)

  $X^{n+1} = X^n \triangledown F(X^n)$ when $F(F(X^n)) \not\sqsubseteq F(X^n)$

  $X^{n+1} = F(X^n)$ when $F(F(X^n)) \sqsubseteq F(X^n)$

- Widening $\triangledown$, two independent *hypotheses*:

- $Y \sqsubseteq X \triangledown Y$ (*extrapolation*)

- Enforces *convergence* of increasing iterates with widening (to a limit X^{ℓ})

The oldest widenings

- Primitive widening [1,2]

```
(x ∇̄ y) = cas x ∈ Va, y ∈ Va dans
   ⊢ □, ? => y ;
   ⊢ ?, □ => x ;
   ⊢ [n1,m1],[n2,m2] =>
         [si n2 < n1 alors -∞ sinon n1 fsi ;
            si m2 > m1 alors +∞ sinon m1 fsi] ;
   fincas ;
```

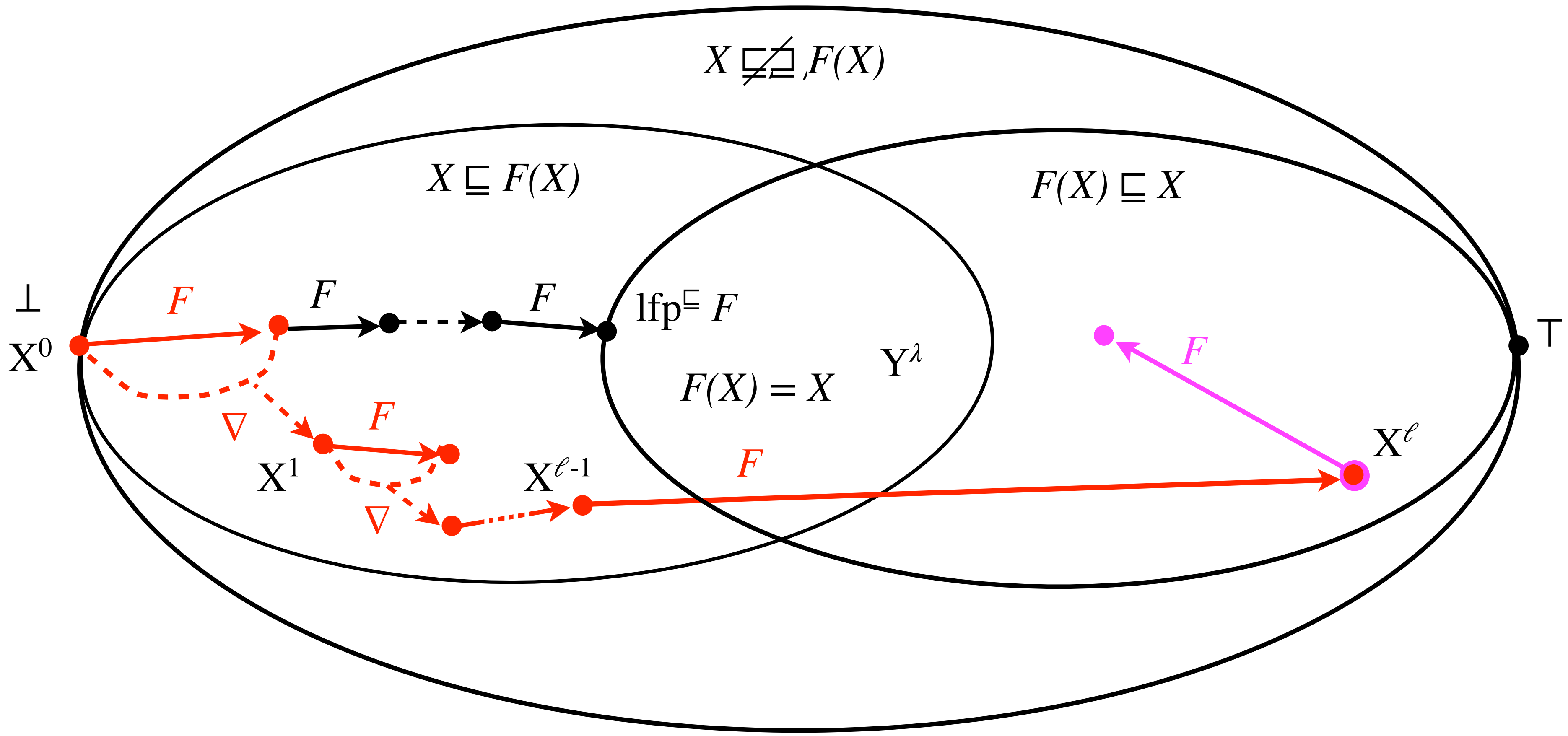
$$[a_1, b_1]\ \bar{\bar{\nabla}}\ [a_2, b_2] =$$
$$[\underline{\text{if}}\ a_2 < a_1\ \underline{\text{then}}\ -\infty\ \underline{\text{else}}\ a_1\ \underline{\text{fi}},$$
$$\underline{\text{if}}\ b_2 > b_1\ \underline{\text{then}}\ +\infty\ \underline{\text{else}}\ b_1\ \underline{\text{fi}}]$$

- Widening with thresholds [3]

$$\forall x \in \bar{L}_2,\ \bot\ \nabla_2(j)\ x = x\ \nabla_2(j)\ \bot = x$$
$$[l_1, u_1]\ \nabla_2(j)\ [l_2, u_2]$$
$$= [\textit{if } 0 \leq l_2 < l_1 \textit{ then } 0 \textit{ elsif } l_2 < l_1 \textit{ then } -b-1 \textit{ else } l_1 \textit{ fi},$$
$$\textit{if } u_1 < u_2 \leq 0 \textit{ then } 0 \textit{ elsif } u_1 < u_2 \textit{ then } b \textit{ else } u_1 \textit{ fi}]$$

[1] Patrick Cousot, Radhia Cousot: Vérification statique de la cohérence dynamique des programmes, Rapport du contrat IRIA-SESORI No 75-032, 23 septembre 1975.
[2] Patrick Cousot, Radhia Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. POPL 1977: 238-252
[3] Patrick Cousot, Semantic foundations of program analysis, Ch. 10 of Program flow analysis: theory and practice, N. Jones & S. Muchnich (eds), Prentice Hall, 1981.

Extrapolation with widening



Widenings are not increasing

- A well-known fact

$[1,1] \subseteq [1,2]$ but $[1,1] \nabla [1,2] = [1,\infty] \not\subseteq [1,2] \nabla [1,2] = [1,2]$

- A widening cannot both:
- Be increasing in its first parameter
- Enforce termination of the iterates
- Avoid useless over-approximations as soon as a solution is found(*)

(*) A counter-example is $x \nabla y = \top$

Narrowing

Interpolation with narrowing

- $Y^0 = X^\ell$ (decreasing iterates with narrowing)

  $Y^{n+1} = Y^n \mathbin{\Delta} F(Y^n)$ when $F(F(Y^n)) \sqsubset F(Y^n)$

  $Y^{n+1} = F(Y^n)$ when $F(F(Y^n)) = F(Y^n)$

- Narrowing Δ, two independent *hypotheses*:
- $Y \sqsubseteq X \implies Y \sqsubseteq X \mathbin{\Delta} Y \sqsubseteq X$ (*interpolation*)
- Enforces *convergence* of decreasing iterates with narrowing (to a limit Y^λ)

The oldest narrowing

- [2]

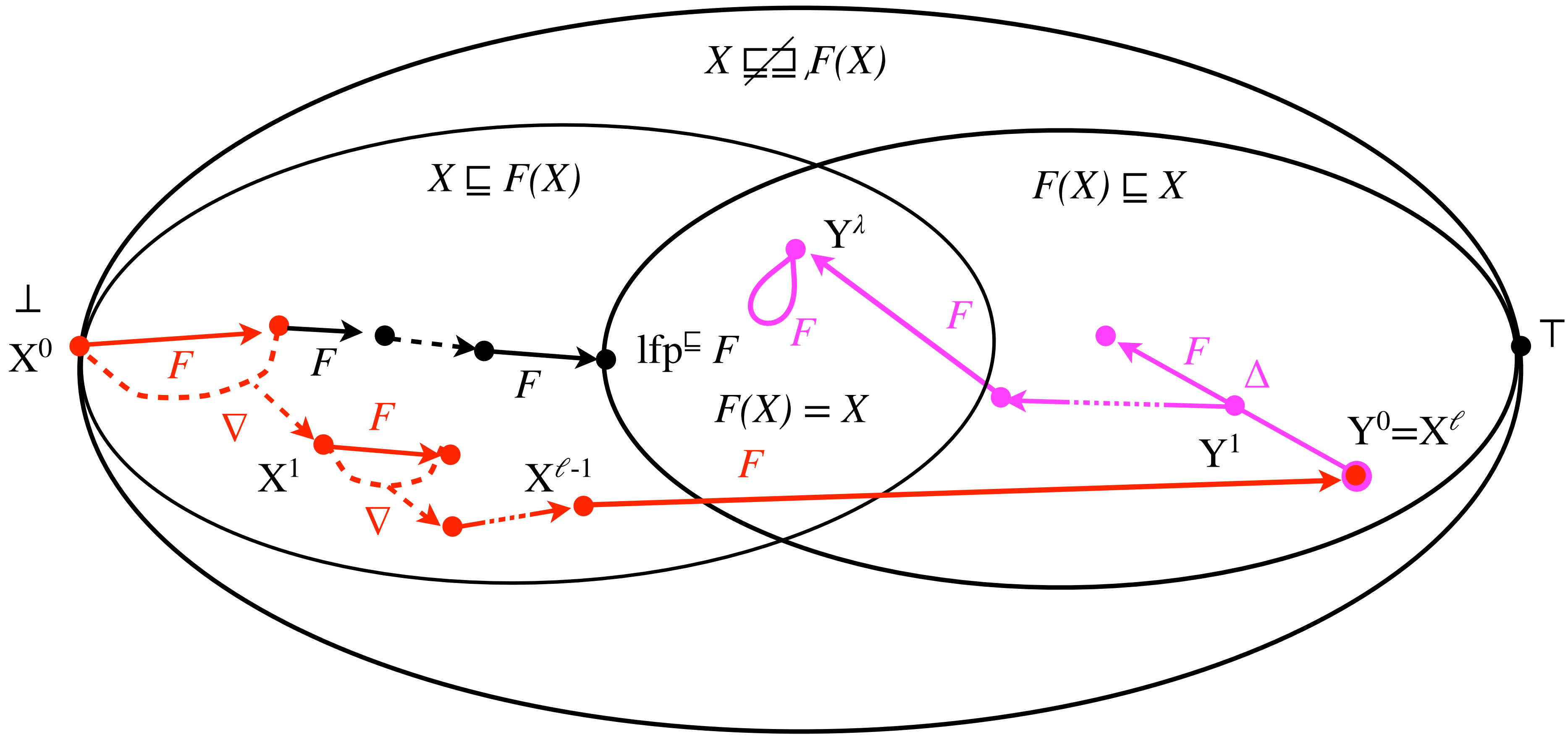
$$[a_1,b_1]\ \bar{\Delta}\ [a_2,b_2] =$$

$$[\underline{\text{if}}\ a_1 = -\infty\ \underline{\text{then}}\ a_2\ \underline{\text{else}}\ \text{MIN}\ (a_1,a_2),$$

$$\underline{\text{if}}\ b_1 = +\infty\ \underline{\text{then}}\ b_2\ \underline{\text{else}}\ \text{MAX}\ (b_1,b_2)]$$

[2] Patrick Cousot, Radhia Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. POPL 1977: 238-252

Interpolation with narrowing



Could stop when $F(X) \not\sqsubseteq X \wedge F(F(X)) \sqsubseteq F(X)$ but not the current practice.

Duality

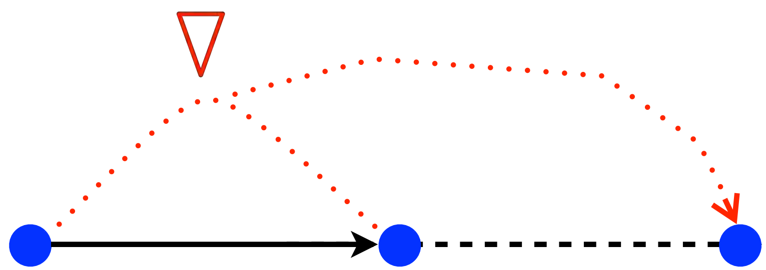
Duality

-

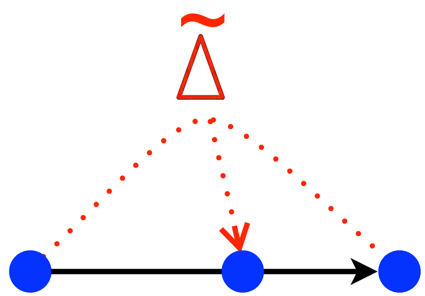
	Convergence above the limit	Convergence below the limit
Increasing iteration	Widening ∇	Dual-narrowing $\widetilde{\Delta}$
Decreasing iteration	Narrowing Δ	Dual widening $\widetilde{\nabla}$

Extrapolators (∇, $\widetilde{\nabla}$) and interpolators (Δ, $\widetilde{\Delta}$)

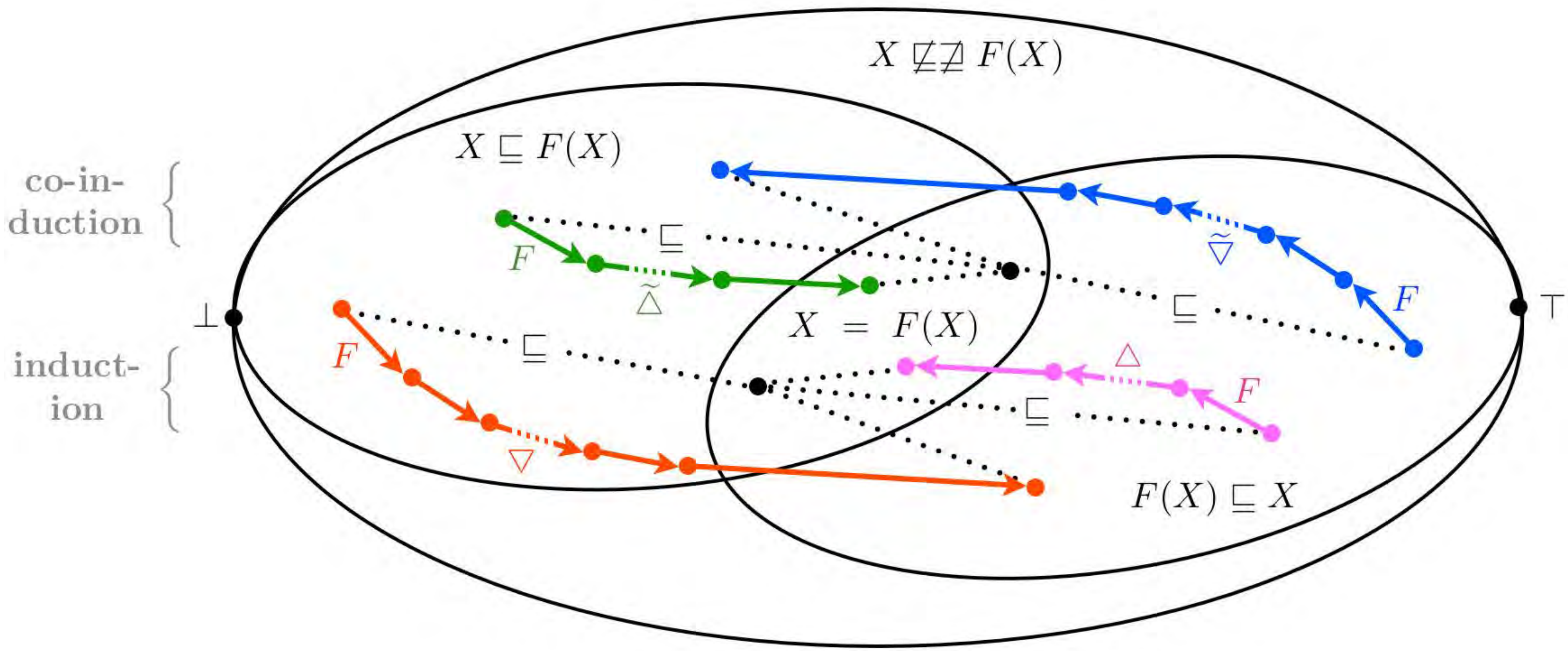
- **Extrapolators**:



- **Interpolators**:



Extrapolators, Interpolators, and Duals



Multi-step extrapolators/interpolators

- The extrapolators/interpolators can be on
  - the last two iterates
  - a bounded number of previous iterates
  - all previous iterates
- Examples:
  - loop unrolling
  - delayed widening
  - etc

Dual narrowing

Interpolation with dual narrowing

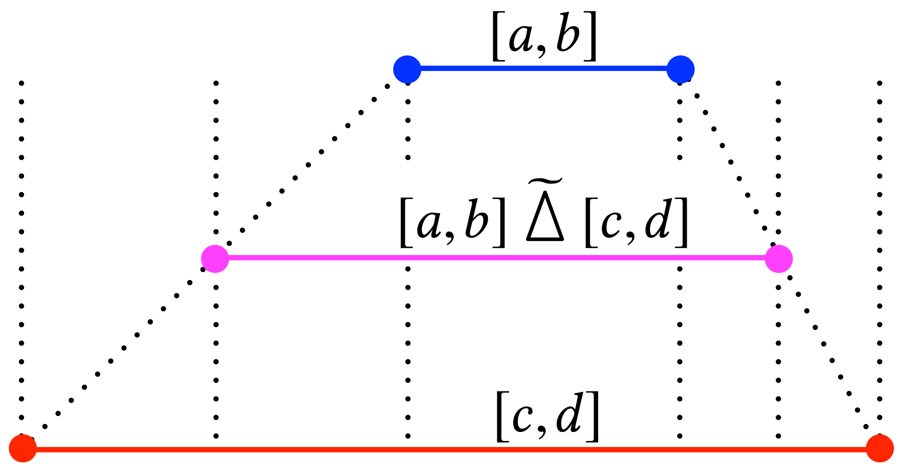
- $Z^0 = \bot$ (increasing iterates with dual-narrowing)

  $Z^{n+1} = F(Z^n) \mathbin{\widetilde{\Delta}} Y^\lambda$ when $F(F(Z^n)) \not\sqsubseteq F(Z^n)$

  $Z^{n+1} = F(Z^n)$ when $F(F(Z^n)) \sqsubseteq F(Z^n)$

- Dual-narrowing $\widetilde{\Delta}$, two independent *hypotheses*:
  - $X \sqsubseteq Y \implies X \sqsubseteq Y \mathbin{\widetilde{\Delta}} X \sqsubseteq Y$ (*interpolation*)
  - Enforces *convergence* of increasing iterates with dual-narrowing

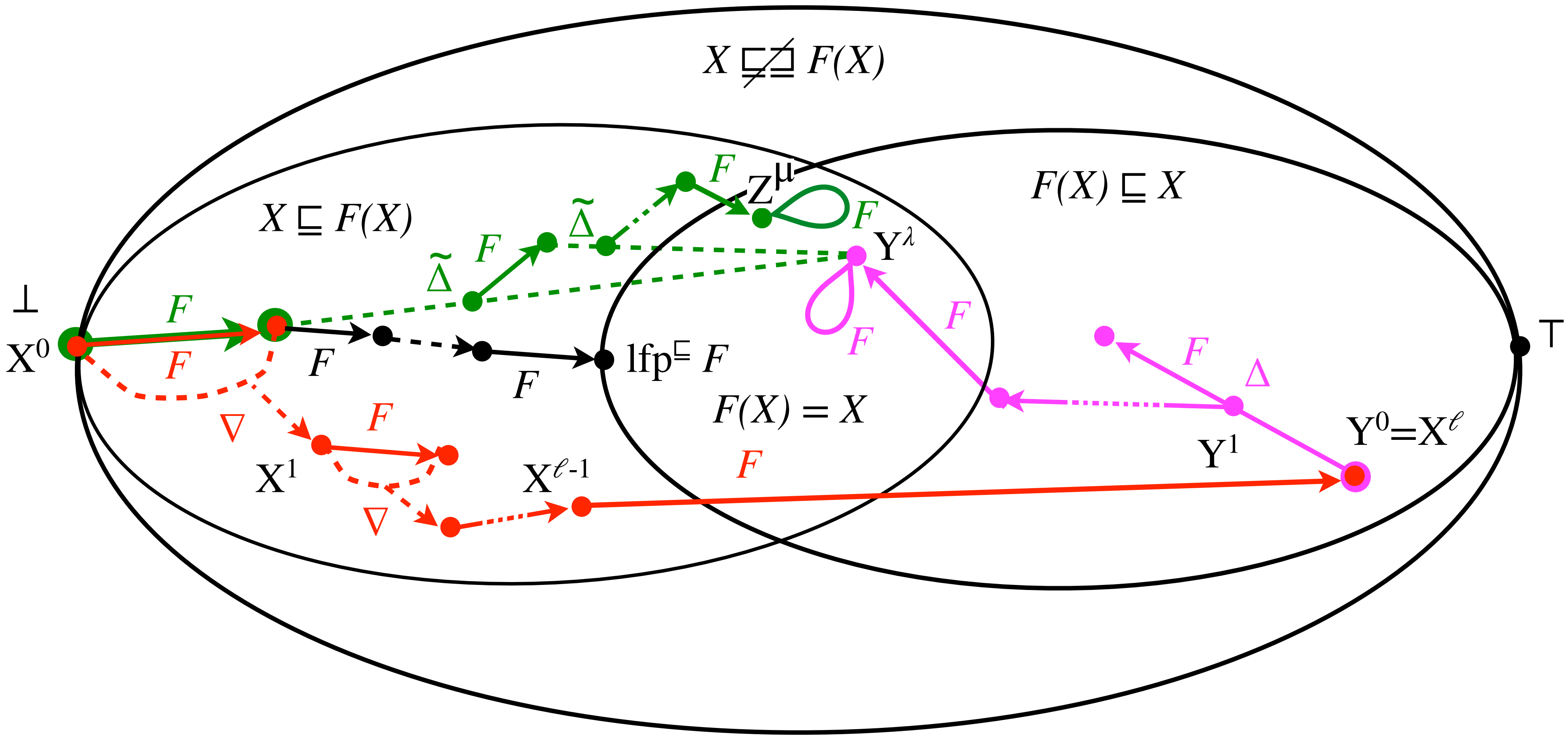
Example of dual-narrowing

- 

- $[a,b] \,\widetilde{\Delta}\, [c,d] \triangleq [\llparenthesis c = -\infty \mathrel{?} a \mathrel{\mathring{\mathord{\circ}}} \lfloor (a+c)/2 \rfloor \rrparenthesis, \llparenthesis d = \infty \mathrel{?} b \mathrel{\mathring{\mathord{\circ}}} \lceil (b+d)/2 \rceil \rrparenthesis]$

- The first method we tried in the late 70's with Radhia
  - Slow
  - Does not easily generalize (e.g. to pointer analysis)

Interpolation with dual-narrowing



- Refine widening/narrowing iterations Y^{λ}
- Refine a user-defined specification (Craig interpolation)

Craig interpolation

- Craig interpolation:

  Given $\mathrm{P} \Longrightarrow \mathrm{Q}$ find I such that $\mathrm{P} \Longrightarrow \mathrm{I} \Longrightarrow \mathrm{Q}$ with

  $\mathrm{var}(\mathrm{I}) \subseteq \mathrm{var}(\mathrm{P}) \cap \mathrm{var}(\mathrm{Q})$

  is a dual narrowing (already observed by Vijay D'Silva and Leopold Haller as a narrowing [indeed inversed narrowing!])

- May not be unique

- May not terminate

Relationship between narrowing and dual-narrowing

- $\widetilde{\Delta} = \Delta^{-1}$
- $Y \sqsubseteq X \implies Y \sqsubseteq X \mathbin{\Delta} Y \sqsubseteq X$ (narrowing)
- $Y \sqsubseteq X \implies Y \sqsubseteq Y \mathbin{\widetilde{\Delta}} X \sqsubseteq X$ (dual-narrowing)

Note: effectiveness and termination conditions may be different

Bounded widening

Dual-narrowing versus bounded widening

- Dual-narrowing $\widetilde{\Delta}$:

  $F(X) \sqsubseteq B \Longrightarrow F(X) \sqsubseteq F(X) \;\widetilde{\Delta}\; B \sqsubseteq B$

  Induction on F(X) and B

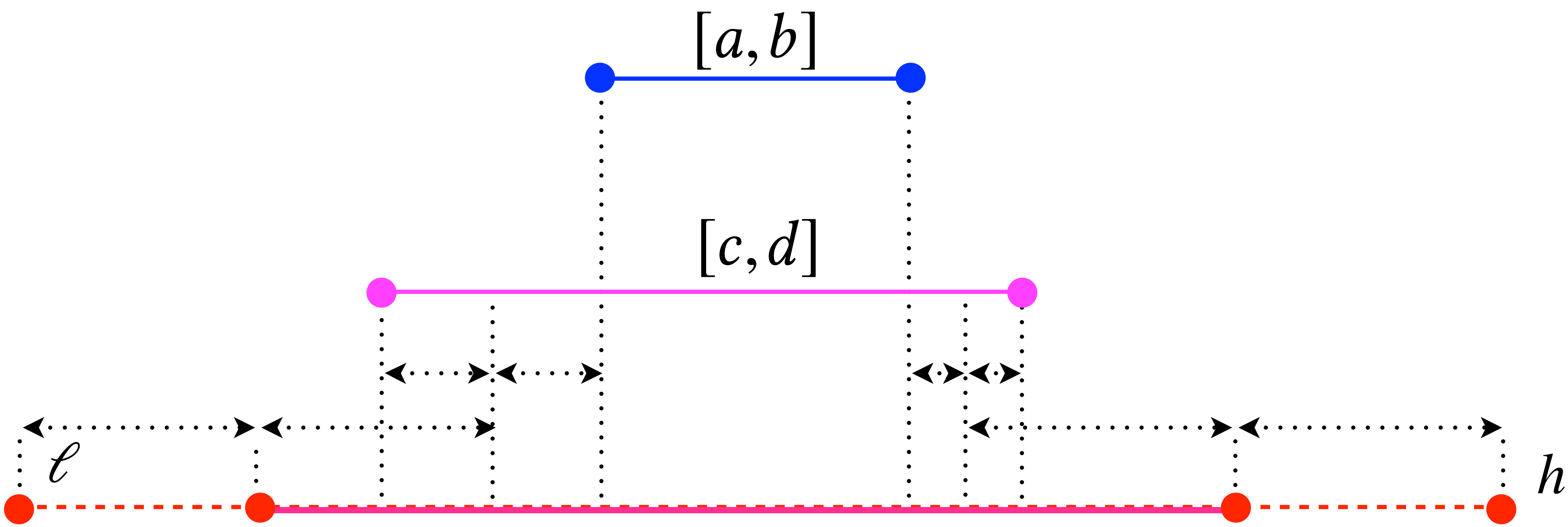
- Bounded widening ∇_B:

  $X \sqsubseteq F(X) \sqsubseteq B \Longrightarrow F(X) \sqsubseteq X \;\nabla_B\; F(X) \sqsubseteq B$

  Induction on X, F(X), and B

Example of widenings (cont'd)

- Bounded widening (in $[\ell, h]$):



$$[a,b] \ \nabla_{[\ell,h]} \ [c,d] \triangleq [\frac{c+a-2\ell}{2}, \frac{b+d+2h}{2}]$$

Soundness

Soundness

- Fixpoint approximation soundness theorems can be expressed with minimalist hypotheses (*):
- No need for complete lattices, complete partial orders (CPO's):
  - The concrete domain is a poset
  - The abstract domain is a pre-order
  - The concretization is defined for the abstract iterates only.

(*) Patrick Cousot. Abstracting Induction by Extrapolation and Interpolation In Deepak D'Souza, Akash Lal, and Kim Guldstrand Larsen (Eds), *16th International Conference on Verification, Model Checking, and Abstract Interpretation*, Mumbai, India, January 12—14, 2015. Lecture Notes in Computer Science, vol. 8931, pp. 19—42, © Springer 2015.

Soundness (cont'd)

- No need for increasingness/monotony hypotheses for fixpoint theorems (Tarski, Kleene, etc)
  - The concrete transformer is increasing and the limit of the iterations does exist in the concrete domain
  - No monotonicity hypotheses on the abstract transformer (no need for fixpoints in the abstract)
  - Soundness hypotheses on the extrapolators/interpolators with respect to the concrete
- In addition, the independent termination hypotheses on the extrapolators/interpolators ensure convergence in finitely many steps

Conclusion

The challenge of verification

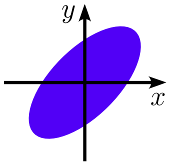
- Infer the inductive argument
- Without deep knowledge about the program (e.g. very precise, quasi-inductive, quasi-strong enough specification)
- Scale

Infer the abstract inductive argument

```
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;
void filter () {
  static float E[2], S[2];
  if (INIT) { S[0] = X; P = X; E[0] = X; }
  else { P = (((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
             + (S[0] * 1.5)) - (S[1] * 0.7)); }
  E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
  /* S[0], S[1] in ???????????????????????????     */
}
void main () { X = 0.2 * X + 5; INIT = TRUE;
  while (1) {
    X = 0.9 * X + 35; /* simulated filter input */
    filter (); INIT = FALSE; }
}
```

Infer the abstract inductive argument

```
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;
void filter () {
  static float E[2], S[2];
  if (INIT) { S[0] = X; P = X; E[0] = X; }
  else { P = (((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
              + (S[0] * 1.5)) - (S[1] * 0.7)); }
  E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
  /* S[0], S[1] in [-1327.02698354, 1327.02698354] */
}
void main () { X = 0.2 * X + 5; INIT = TRUE;
  while (1) {
    X = 0.9 * X + 35; /* simulated filter input */
    filter (); INIT = FALSE; }
}
```



Extrapolation/Interpolation

- Abstract interpretation in infinite domains is traditionally by iteration with widening/narrowing.
- We have shown how to use iteration with dual-narrowing.
- These ideas of the 70's generalize Craig interpolation from logic to arbitrary abstract domains.
- Can be used to improve precision when a fixpoint is reached after the widening/narrowing iterations

The End, Thank You