

# Abstract Induction

ETH Zürich  
October 2–3, 2015

Patrick Cousot

[pcousot@cs.nyu.edu](mailto:pcousot@cs.nyu.edu) [cs.nyu.edu/~pcousot](http://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:

## (I) 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 ( $\Rightarrow$  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++`  $\longrightarrow \{\perp, [0,0], [0,1], [-\infty, \infty]\}$

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

...

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

...

- 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**:

$$\langle \wp(\mathcal{D} \llbracket P \rrbracket), \subseteq \rangle \xrightleftharpoons[\alpha \llbracket P \rrbracket]{\gamma \llbracket P \rrbracket} \langle \mathcal{A} \llbracket P \rrbracket, \sqsubseteq \rangle$$

- Calculate the **abstract semantics**:

$$S^\# \llbracket P \rrbracket = \alpha \llbracket P \rrbracket (\{ S \llbracket P \rrbracket \}) \quad \text{exact abstraction}$$

$$S^\# \llbracket P \rrbracket \sqsupseteq \alpha \llbracket P \rrbracket (\{ S \llbracket P \rrbracket \}) \quad \text{approximate abstraction}$$

- Soundness** (by construction):

$$\forall P \in \mathbb{L}: \forall Q \in \mathcal{A}: S^\# \llbracket P \rrbracket \sqsubseteq Q \implies S \llbracket P \rrbracket \in \gamma \llbracket P \rrbracket (Q)$$

# Refinement: good news

- **Problem**: how to prove a valid abstract property  $\alpha(\{\text{lfp } F \llbracket P \rrbracket\}) \sqsubseteq Q$  when  $\alpha \circ F \sqsubseteq F^\# \circ \alpha$  but  $\text{lfp } F^\# \llbracket P \rrbracket \not\sqsubseteq Q$ ? (i.e. strongest inductive argument too weak)
- It is **always** possible to refine  $\langle \mathcal{A}, \sqsubseteq \rangle$  into a most abstract more precise abstraction  $\langle \mathcal{A}', \sqsubseteq' \rangle$  such that

$$\langle \wp(\mathcal{D}), \sqsubseteq \rangle \xrightleftharpoons[\alpha']{\gamma'} \langle \mathcal{A}', \sqsubseteq' \rangle$$

and  $\alpha' \circ F = F' \circ \alpha$  with  $\text{lfp } F' \llbracket P \rrbracket \sqsubseteq' \alpha' \circ \gamma(Q)$

(thus proving  $\text{lfp } F \llbracket P \rrbracket \in \gamma'(Q)$  which implies  $\text{lfp } F \llbracket P \rrbracket \in \gamma(Q)$ )

---

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

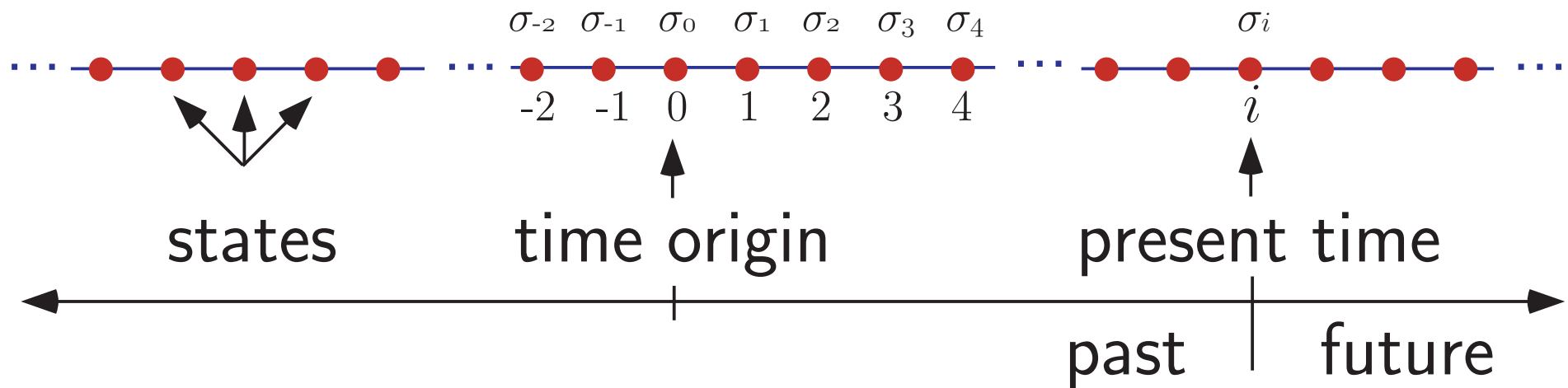
# Refinement: bad news

- 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:



# Example of intrinsic approximate refinement

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

$\varphi ::=$	$\sigma_S$	$S \in \wp(\mathbb{S})$	state predicate
	$\pi_t$	$t \in \wp(\mathbb{S} \times \mathbb{S})$	transition predicate
	$\oplus \varphi_1$		next
	$\varphi_1^\sim$		reversal
	$\varphi_1 \vee \varphi_2$		disjunction
	$\neg \varphi_1$		negation
	$X$	$X \in \mathbb{X}$	variable
	$\mu X \cdot \varphi_1$		least fixpoint
	$\nu X \cdot \varphi_1$		greatest fixpoint
	$\forall \varphi_1 : \varphi_2$		universal state closure

# Example of intrinsic approximate refinement

- Consider **universal model-checking abstraction**:

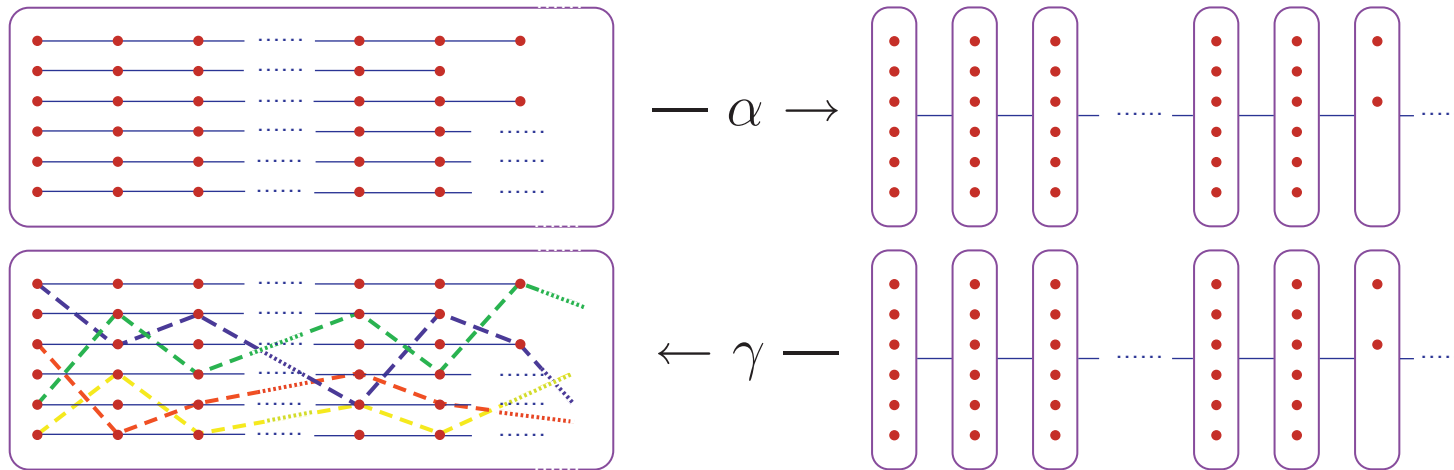
$$\begin{aligned}\text{MC}_M^\forall(\phi) &= \alpha_M^\forall(\llbracket \phi \rrbracket) \in \wp(\text{Traces}) \rightarrow \wp(\text{States}) \\ &= \{s \in \text{States} \mid \forall \langle i, \sigma \rangle \in \text{Traces}_M. (\sigma_i = s) \Rightarrow \\ &\quad \langle i, \sigma \rangle \in \llbracket \phi \rrbracket\}\end{aligned}$$

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 (\*)



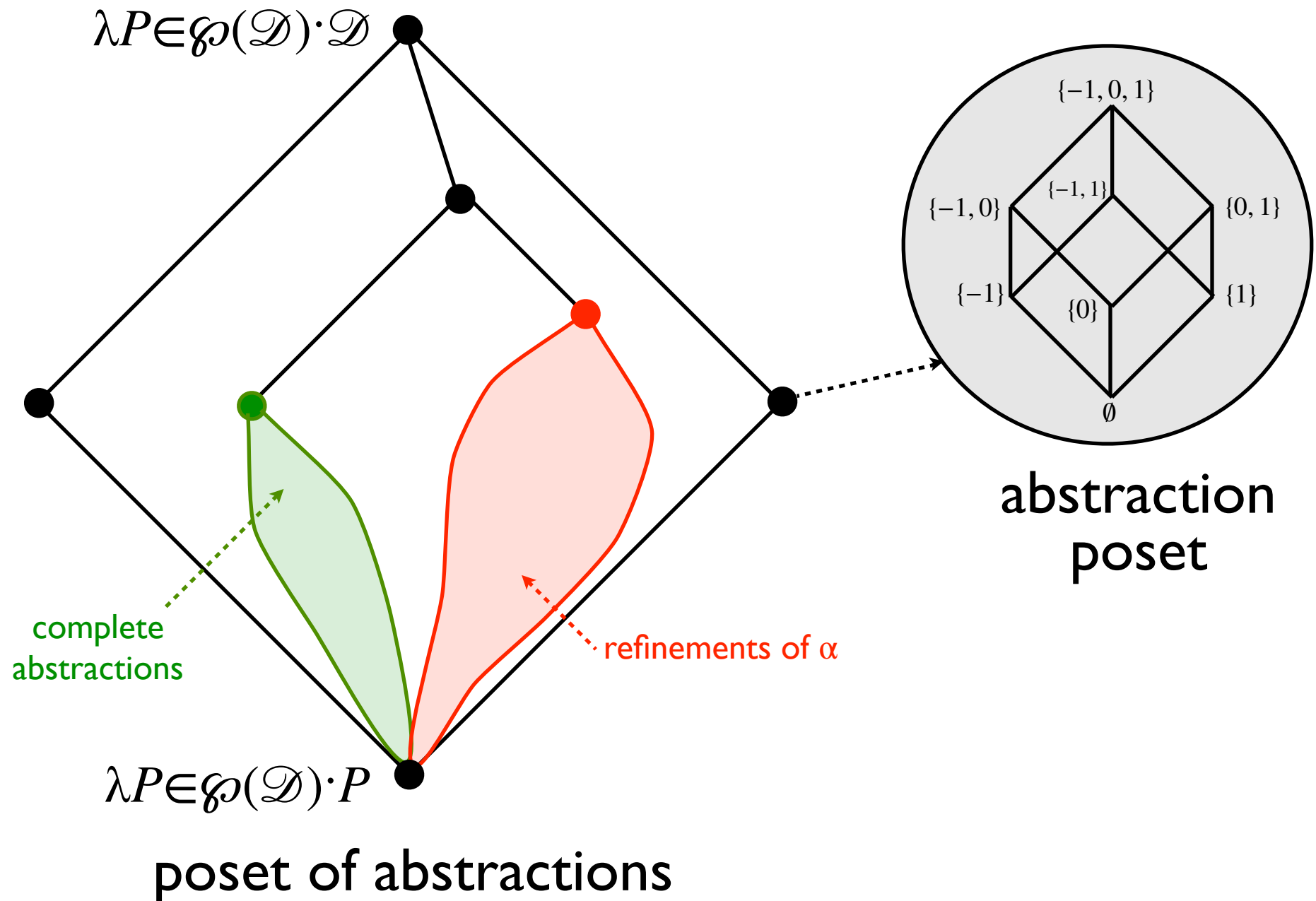
- Any refinement of this abstraction is *incomplete* (but to the infinite past/future trace semantics itself) (\*\*)

---

(\*) 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)

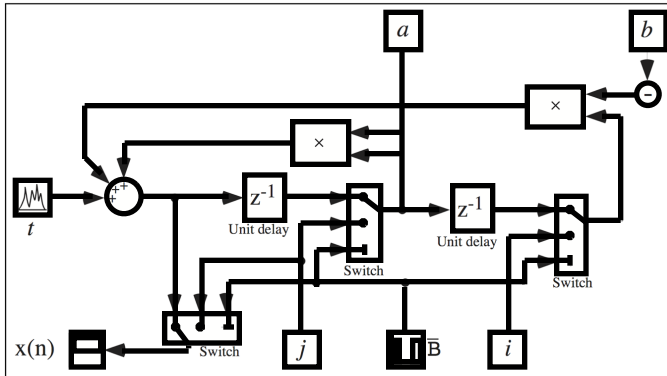
# Intrinsic approximate refinement



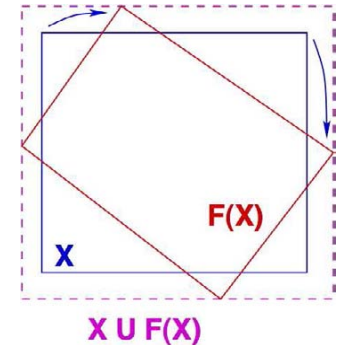
# In general refinement does not terminate

- Example: filter invariant abstraction:

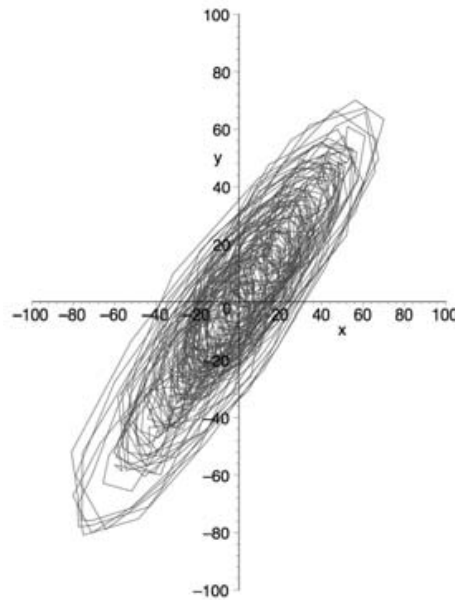
2nd order filter:



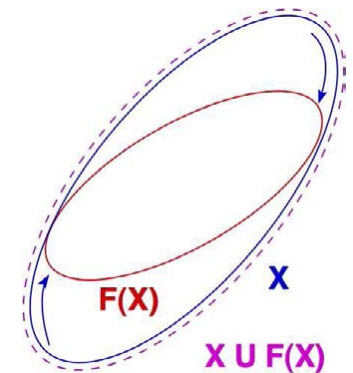
Unstable polyhedral abstraction:



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



Stable ellipsoidal abstraction:



# 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 <sup>(\*)</sup>)

- **Intelligence is needed** for refinement:

e.g. human-driven refinement of Astrée <sup>(\*\*)</sup>

---

(\*) 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<sup>(\*)</sup> abstract domains <sup>(\*\*)</sup>

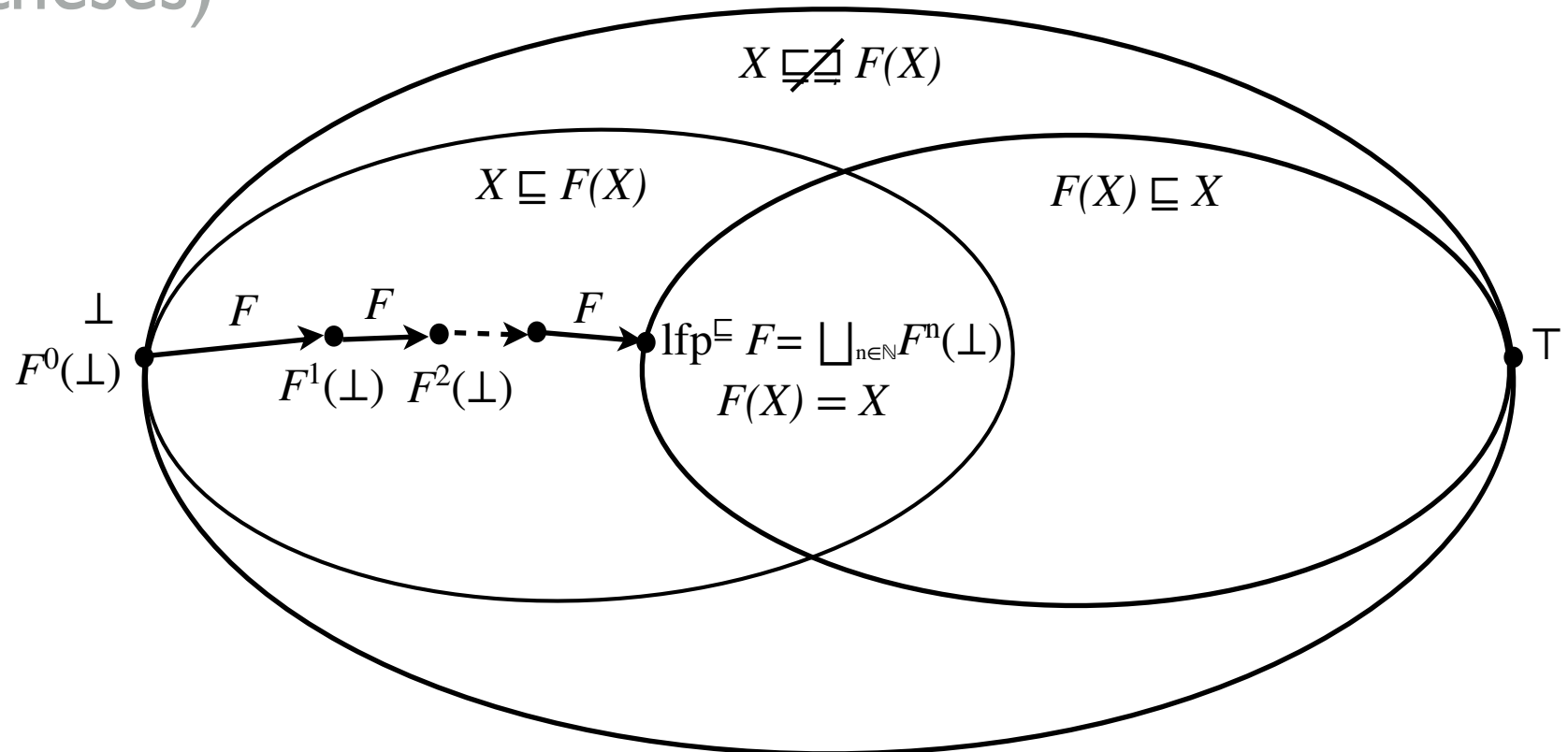
---

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

(\*\*) Or convergence may be guaranteed but too slow.

# Fixpoints

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



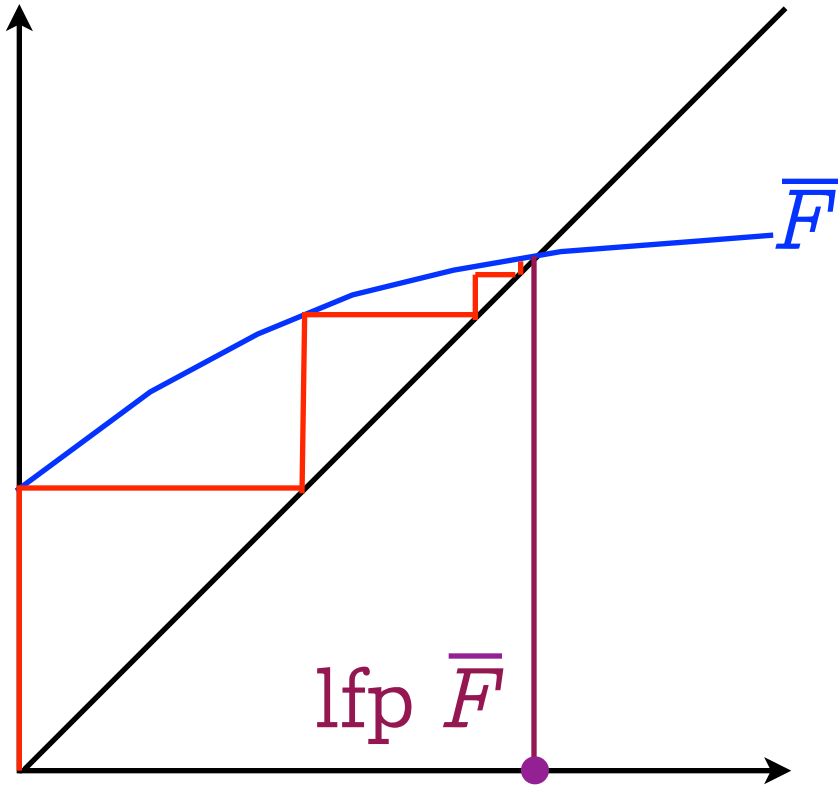
# Convergence criterion

- By Tarski (or variants)

$$F(X) \sqsubseteq X \quad \Rightarrow \quad \text{lfp}^{\sqsubseteq} F \sqsubseteq X$$

# Widening

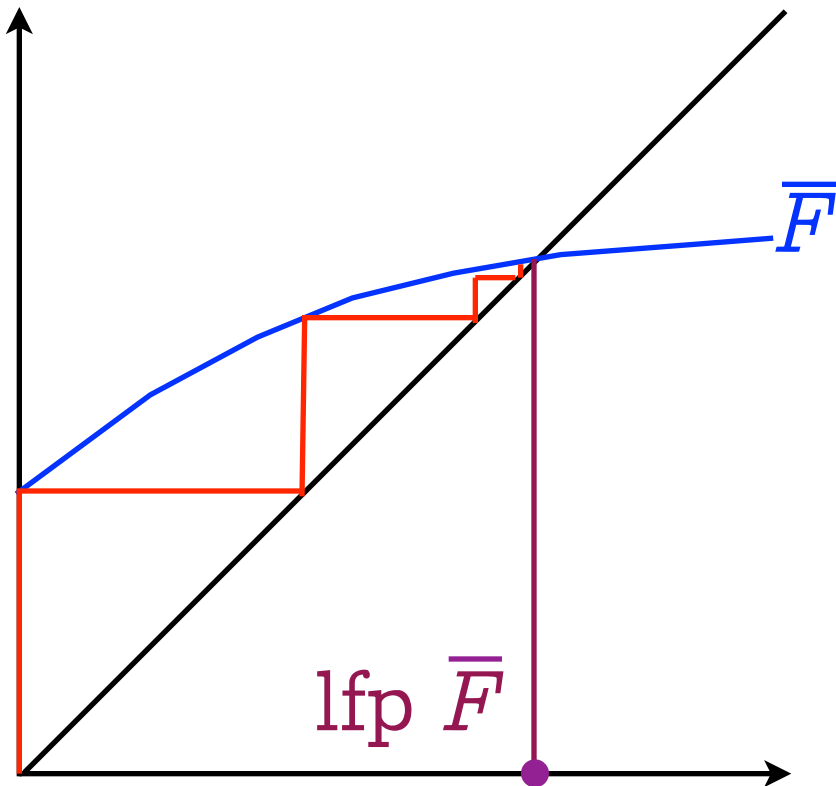
# Convergence acceleration with widening



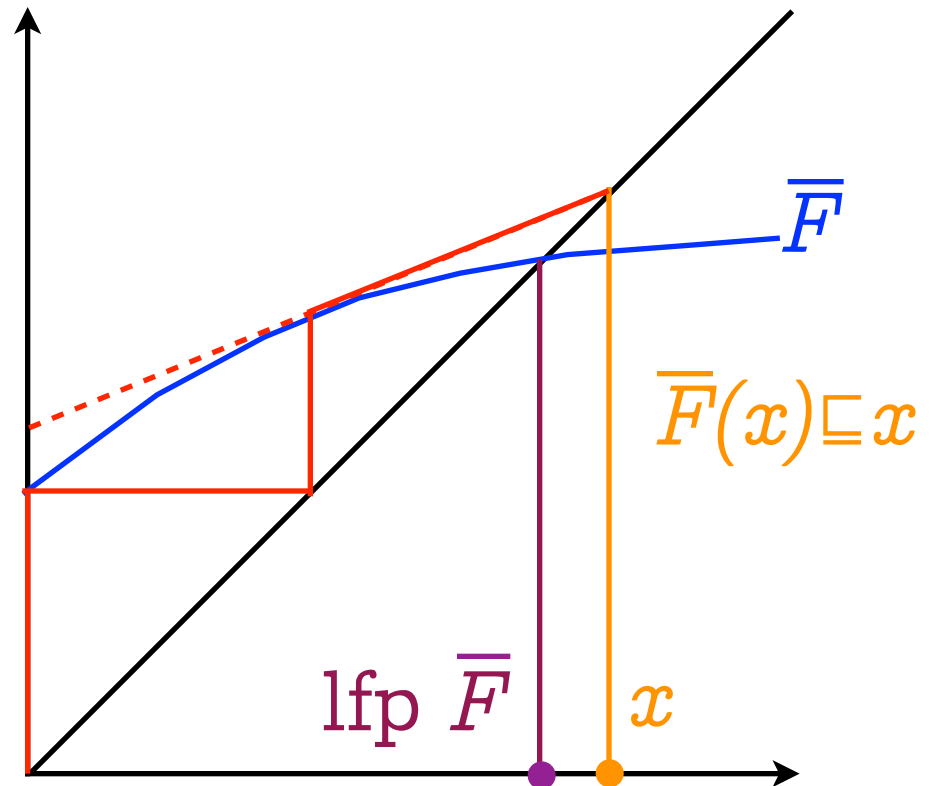
Infinite iteration



# Convergence acceleration with widening



Infinite iteration



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

<sup>(\*)</sup> 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)  
 $X^{n+1} = X^n \nabla F(X^n)$  when  $F(F(X^n)) \not\subseteq F(X^n)$   
 $X^{n+1} = F(X^n)$  when  $F(F(X^n)) \subseteq F(X^n)$
- Widening  $\nabla$ , two independent *hypotheses*:
- $Y \subseteq X \nabla Y$  (*extrapolation*)
- Enforces *convergence* of increasing iterates with widening (to a limit  $X^\ell$ )

# The oldest widenings

- Primitive widening [1,2]

```

(x  $\bar{\nabla}$  y) = cas x  $\in$  Va, y  $\in$  Va dans
|
|  $\square$ , ?  $\Rightarrow$  y ;
| ? ,  $\square$   $\Rightarrow$  x ;
|  $[n_1, m_1], [n_2, m_2] \Rightarrow$ 
|    $[ \underline{\text{si}} \ n_2 < n_1 \ \underline{\text{alors}} \ -\infty \ \underline{\text{sinon}} \ n_1 \ \underline{\text{fsi}} \ ;$ 
|      $\underline{\text{si}} \ m_2 > m_1 \ \underline{\text{alors}} \ +\infty \ \underline{\text{sinon}} \ m_1 \ \underline{\text{fsi}} ] \ ;$ 
| fincas ;
  
```

```

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

- Widening with thresholds [3]

```

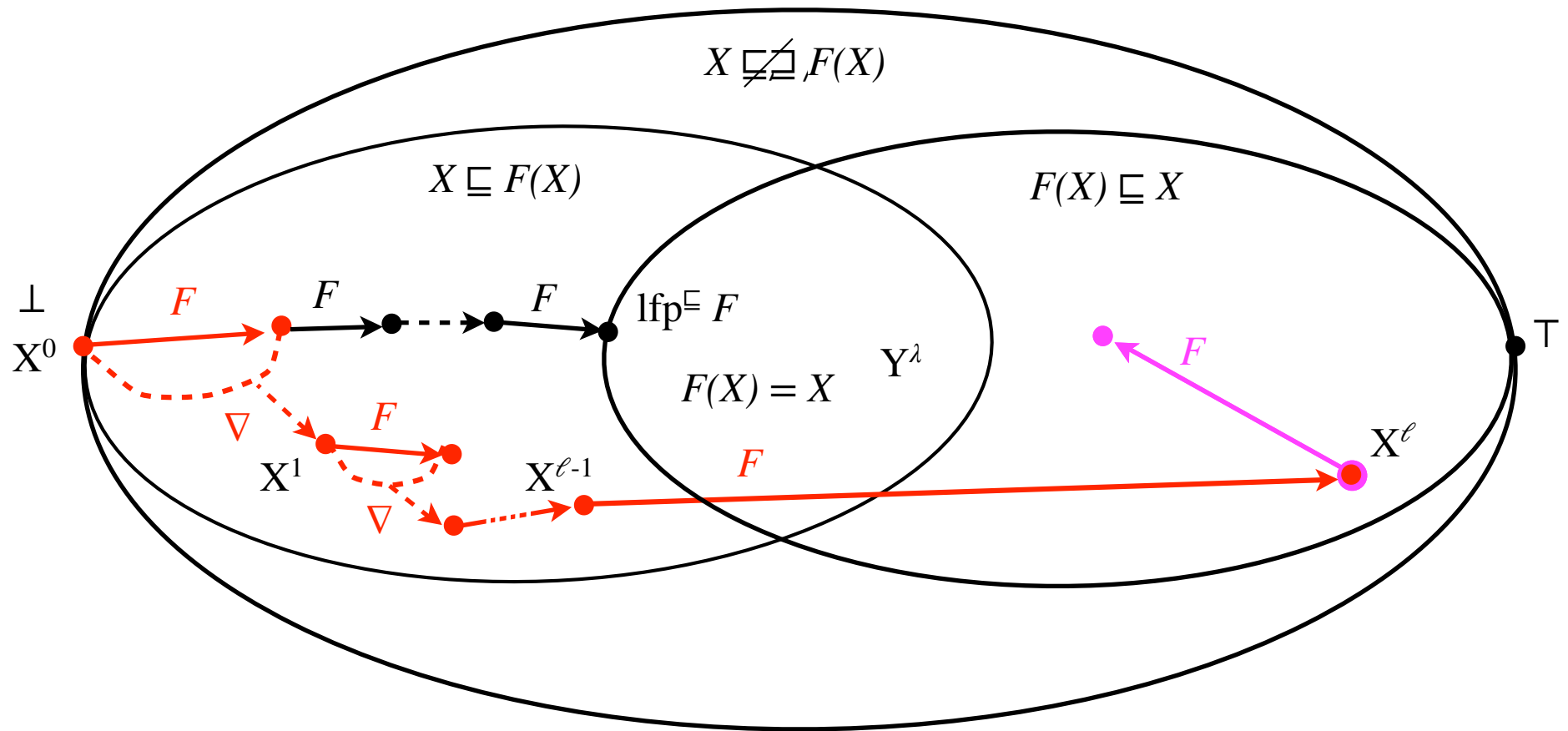
 $\forall x \in \bar{L}_2, \perp \nabla_2(j) x = x \nabla_2(j) \perp = x$ 
 $[l_1, u_1] \nabla_2(j) [l_2, u_2]$ 
 $= [ \text{if } 0 \leq l_2 < l_1 \ \text{then } 0 \ \text{elseif } l_2 < l_1 \ \text{then } -b - 1 \ \text{else } l_1 \ \text{fi},$ 
 $\text{if } u_1 < u_2 \leq 0 \ \text{then } 0 \ \text{elseif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

[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. Muchnick (eds), Prentice Hall, 1981.

# Extrapolation with widening



# Widenings are not increasing

- A **well-known** fact

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

- 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<sup>(\*)</sup>

---

<sup>(\*)</sup> A counter-example is  $x \nabla y = \top$

# Narrowing

# Interpolation with narrowing

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

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

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

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

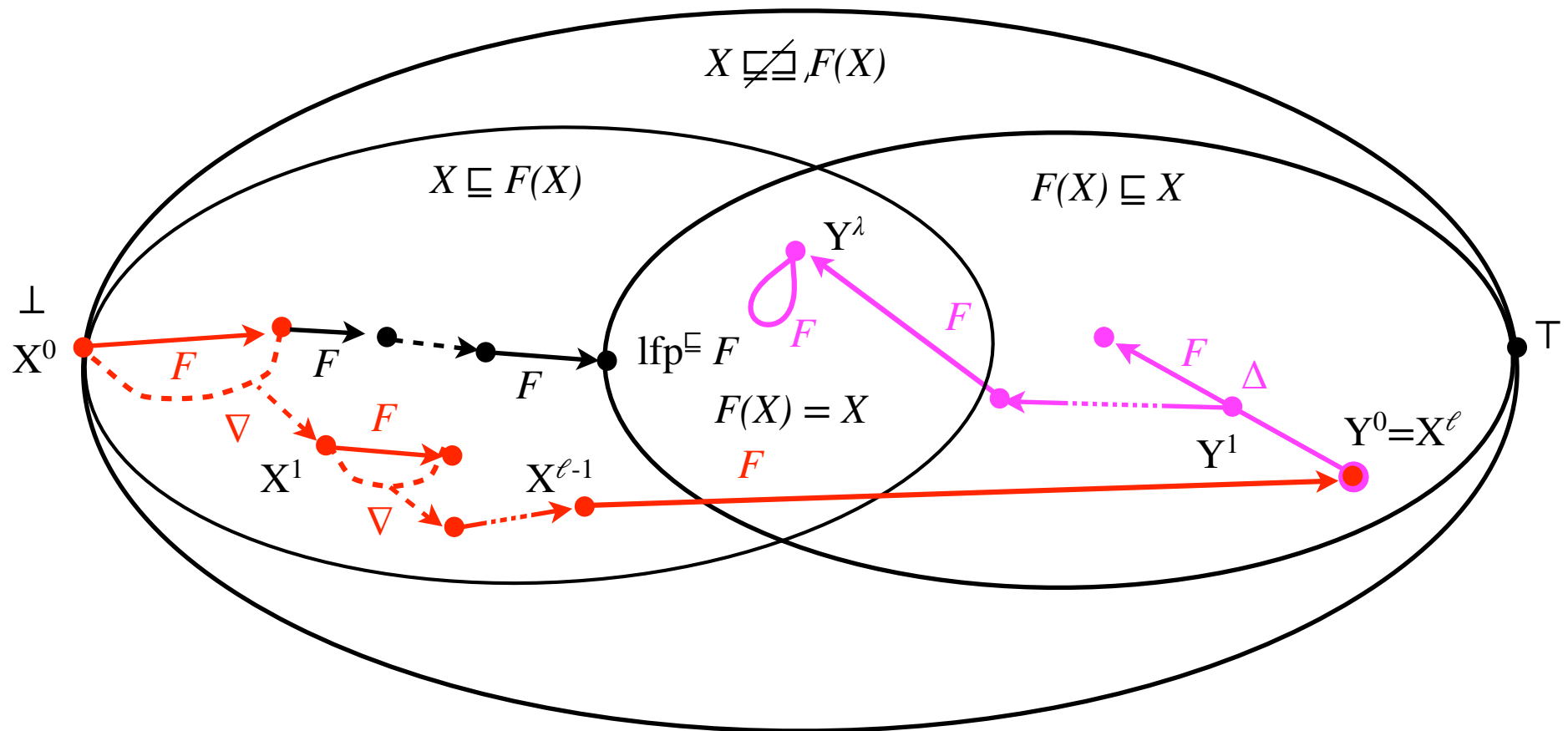
# The oldest narrowing

- [2]

$$\begin{aligned} [a_1, b_1] \bar{\Delta} [a_2, b_2] = \\ [\text{if } a_1 = -\infty \text{ then } a_2 \text{ else MIN } (a_1, a_2), \\ \text{if } b_1 = +\infty \text{ then } b_2 \text{ else MAX } (b_1, b_2)] \end{aligned}$$



# Interpolation with narrowing



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

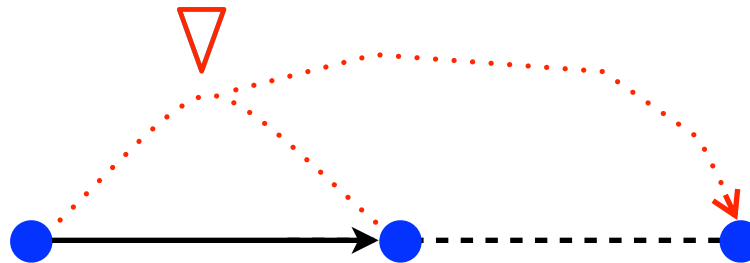
# Duality

# Duality

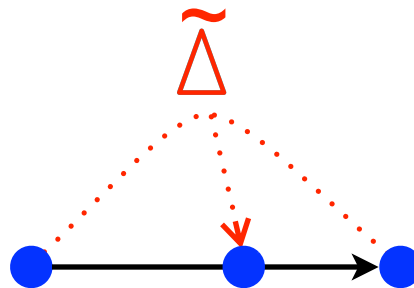
●	Convergence above the limit	Convergence below the limit
Increasing iteration	Widening $\nabla$	Dual-narrowing $\tilde{\Delta}$
Decreasing iteration	Narrowing $\Delta$	Dual widening $\tilde{\nabla}$

Extrapolators ( $\nabla, \tilde{\nabla}$ ) and interpolators ( $\Delta, \tilde{\Delta}$ )

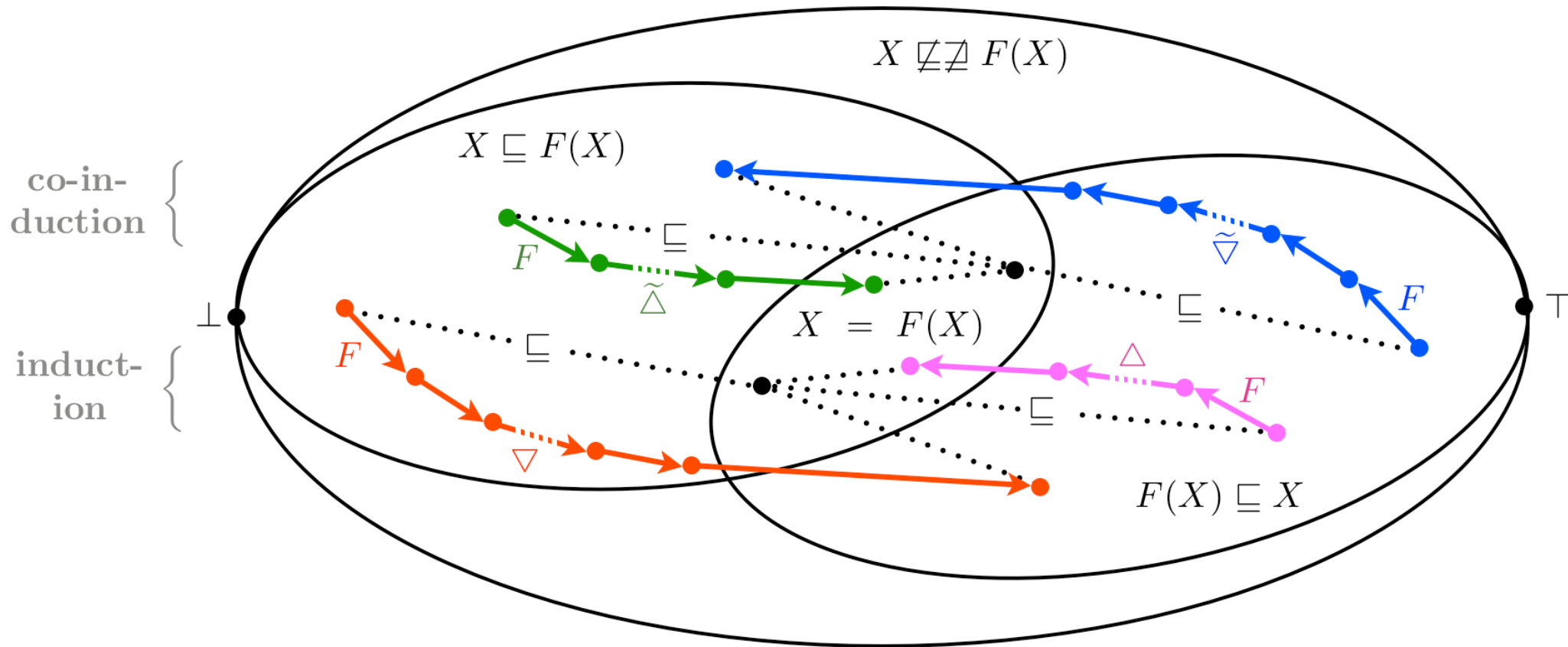
## ● Extrapolators:



## ● 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

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

$$Z^{n+1} = F(Z^n) \tilde{\Delta} Y^\lambda \quad \text{when } F(F(Z^n)) \not\sqsubseteq F(Z^n)$$

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

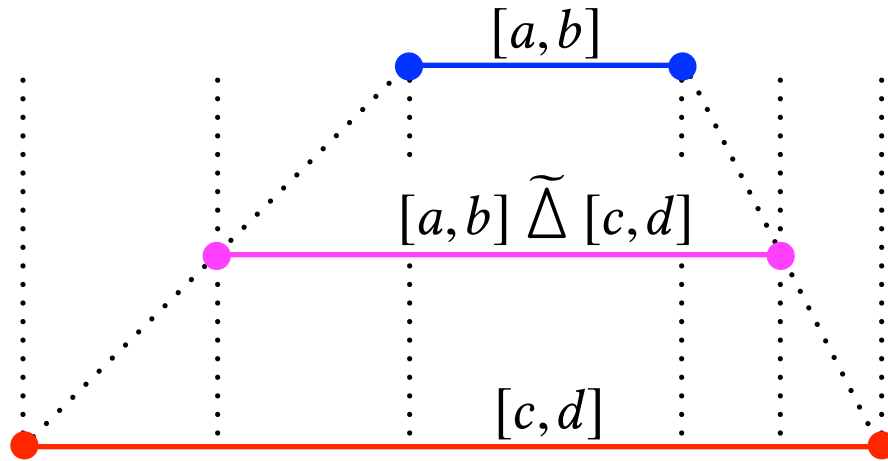
- Dual-narrowing  $\tilde{\Delta}$ , two independent *hypotheses*:

- $X \sqsubseteq Y \implies X \sqsubseteq Y \tilde{\Delta} X \sqsubseteq Y$  (*interpolation*)

- Enforces *convergence* of increasing iterates with dual-narrowing

# Example of dual-narrowing

●



●  $[a, b] \tilde{\Delta} [c, d] \triangleq [(\lfloor c = -\infty \text{ ? } a : \lfloor (a + c)/2 \rfloor), (\lceil d = \infty \text{ ? } b : \lceil (b + d)/2 \rceil)]$

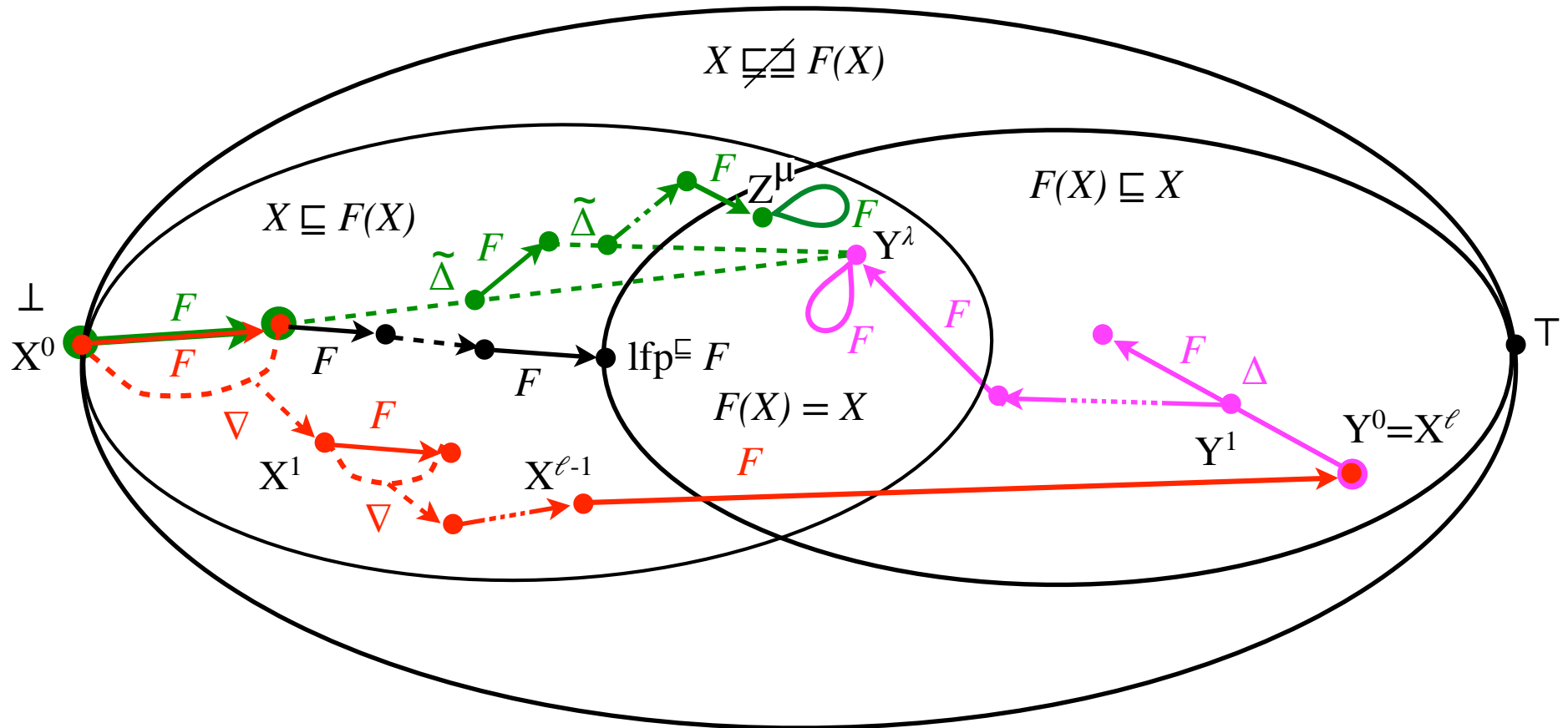
● The first method we tried in the late 70's with Radhia

● Slow

● Does not easily generalize (e.g. to pointer analysis)



# Interpolation with dual-narrowing



- Refine widening/narrowing iterations  $Y^\lambda$
- Refine a user-defined specification (Craig interpolation)

# Craig interpolation

- Craig interpolation:

Given  $P \implies Q$  find  $I$  such that  $P \implies I \implies Q$  with  
 $\text{var}(I) \subseteq \text{var}(P) \cap \text{var}(Q)$

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

- May not be unique
- May not terminate

# Relationship between narrowing and dual-narrowing

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

# Bounded widening

# Dual-narrowing versus bounded widening

- Dual-narrowing:

$$F(X) \sqsubseteq B \implies F(X) \sqsubseteq F(X) \Delta \tilde{B} \sqsubseteq B$$

Induction on  $F(X)$  and  $B$

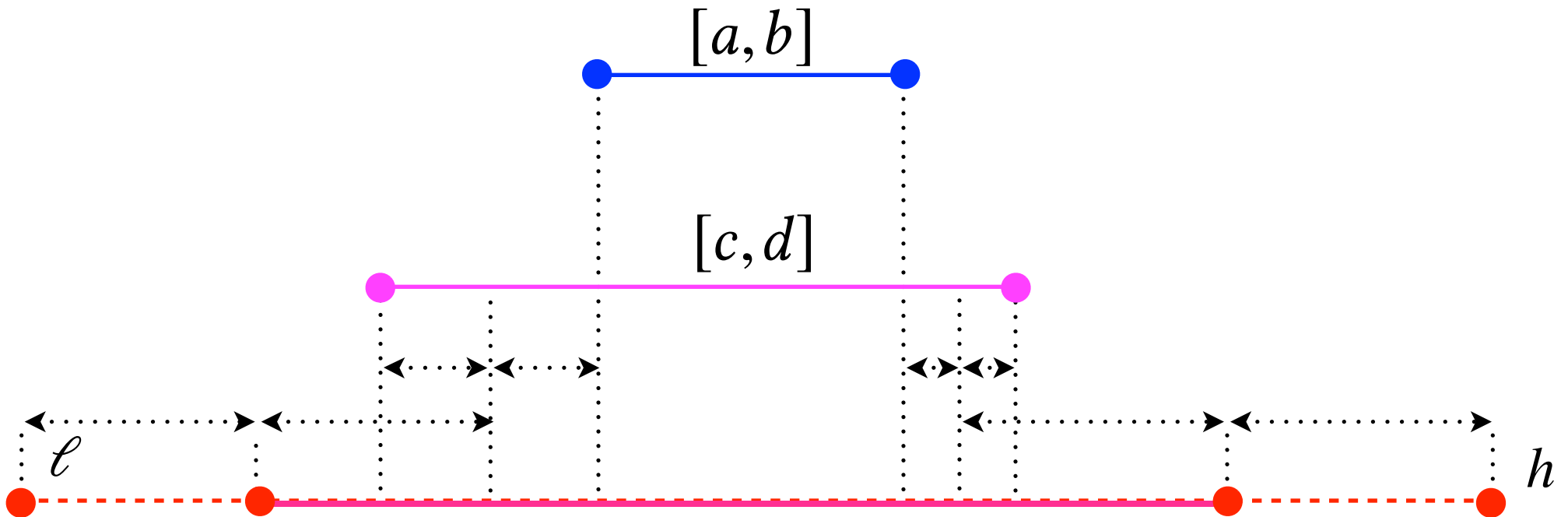
- Bounded widening:

$$X \sqsubseteq F(X) \sqsubseteq B \implies F(X) \sqsubseteq X \nabla_B F(X) \sqsubseteq B$$

Induction on  $X$ ,  $F(X)$ , and  $B$

# Example of widenings (cont'd)

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



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

# 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), *16<sup>th</sup> 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 ?????????????????????????????? */
}

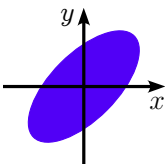
void main () { X = 0.2 * X + 5; INIT = TRUE;
    while (1) {
        X = 0.9 * X + 35; /* simulated filter input */
        filter (); INIT = FALSE; }
}
```

# 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