

Static analysis by abstract interpretation

« A Tutorial on Abstract Interpretation »

Patrick Cousot

École normale supérieure
45 rue d'Ulm
75230 Paris cedex 05, France

Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot

VMCAI'05 Industrial Day

Example of static analysis (input)

```
n := n0;  
i := n;  
while (i <> 0) do  
  j := 0;  
  while (j <> i) do  
    j := j + 1  
  od;  
  i := i - 1  
od
```



Example of static analysis (output)

```

{n0>=0}
  n := n0;
{n0=n,n0>=0}
  i := n;
{n0=i,n0=n,n0>=0}
  while (i <> 0 ) do
    {n0=n,i>=1,n0>=i}
    j := 0;
    {n0=n,j=0,i>=1,n0>=i}
    while (j <> i) do
      {n0=n,j>=0,i>=j+1,n0>=i}
      j := j + 1
      {n0=n,j>=1,i>=j,n0>=i}
    od;
    {n0=n,i=j,i>=1,n0>=i}
    i := i - 1
    {i+1=j,n0=n,i>=0,n0>=i+1}
  od
{n0=n,i=0,n0>=0}

```

Example of static analysis (safety)

```

{n0>=0}
  n := n0;
{n0=n,n0>=0}
  i := n;
{n0=i,n0=n,n0>=0}
  while (i <> 0 ) do
    {n0=n,i>=1,n0>=i}
    j := 0;
    {n0=n,j=0,i>=1,n0>=i}
    while (j <> i) do
      {n0=n,j>=0,i>=j+1,n0>=i}
      j := j + 1
      {n0=n,j>=1,i>=j,n0>=i}
    od;
    {n0=n,i=j,i>=1,n0>=i}
    i := i - 1
    {i+1=j,n0=n,i>=0,n0>=i+1}
  od
{n0=n,i=0,n0>=0}

```

n0 must be initially nonnegative
 (otherwise the program does not
 terminate properly)

← j < n0 so no upper overflow

← i > 0 so no lower overflow



Static analysis by abstract interpretation

Verification: define and prove automatically a **property** of the **possible behaviors** of a complex computer program (example: program semantics);

Abstraction: the reasoning/calculus can be done on an **abstraction** of these behaviors dealing only with those elements of the behaviors related to the considered property;

Theory: abstract interpretation.

— 5 —

Example of static analysis

Verification: absence of runtime errors;

Abstraction: polyhedral abstraction (affine inequalities);

Theory: abstract interpretation.

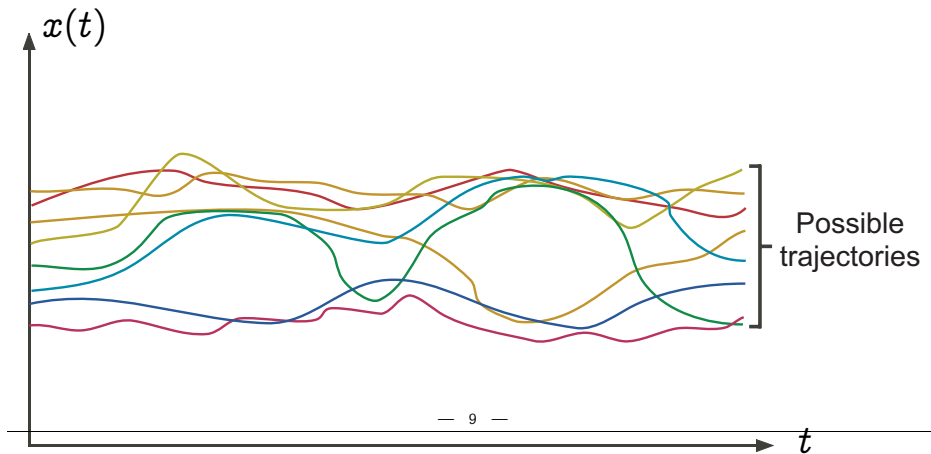
A very informal introduction
to the principles of
abstract interpretation

— 7 —

Semantics

The **concrete semantics** of a program formalizes (is a mathematical model of) the set of all its possible executions in all possible execution environments.

Graphic example: Possible behaviors



Undecidability

- The concrete mathematical semantics of a program is an “infinite” mathematical object, *not computable*;
- All non trivial questions on the concrete program semantics are *undecidable*.

Example: termination

- Assume `termination(P)` would always terminates and returns true iff P always terminates on all input data;
- The following program yields a contradiction

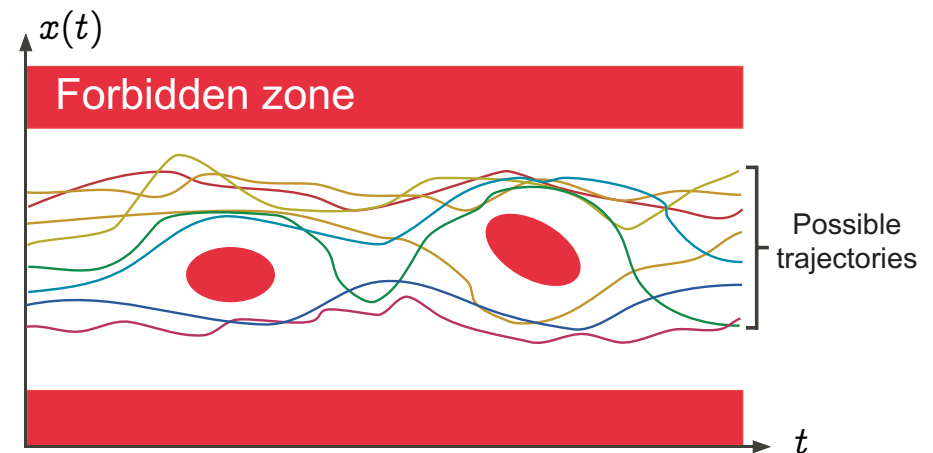
`P ≡ while termination(P) do skip od.`

Graphic example: Safety properties

The *safety properties* of a program express that no possible execution in any possible execution environment can reach an *erroneous state*.

— 11 —

Graphic example: Safety property



Safety proofs

- A **safety proof** consists in proving that the intersection of the program concrete semantics and the forbidden zone is empty;
- **Undecidable** problem (the concrete semantics is not computable);
- Impossible to provide completely automatic answers with finite computer resources and neither human interaction nor uncertainty on the answer¹.

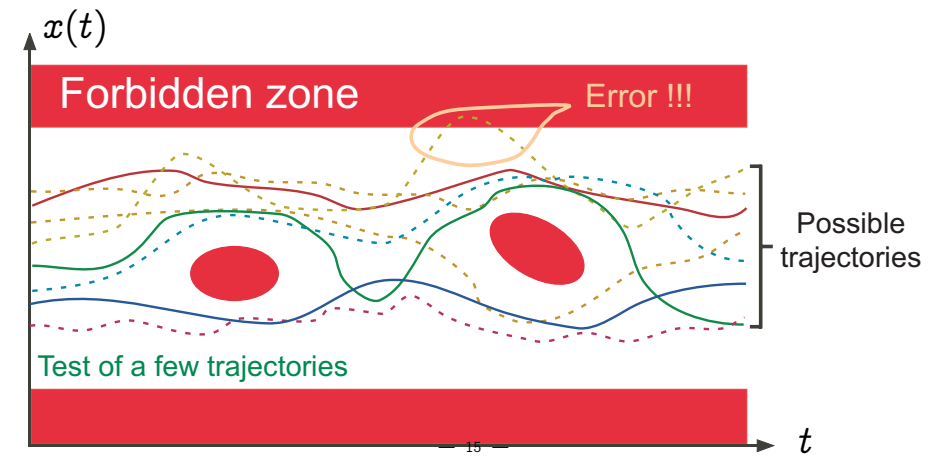
— 13 —

Test/debugging

- consists in considering a subset of the possible executions;
- not a correctness proof;
- **absence of coverage** is the main problem.

¹ e.g. probabilistic answer.

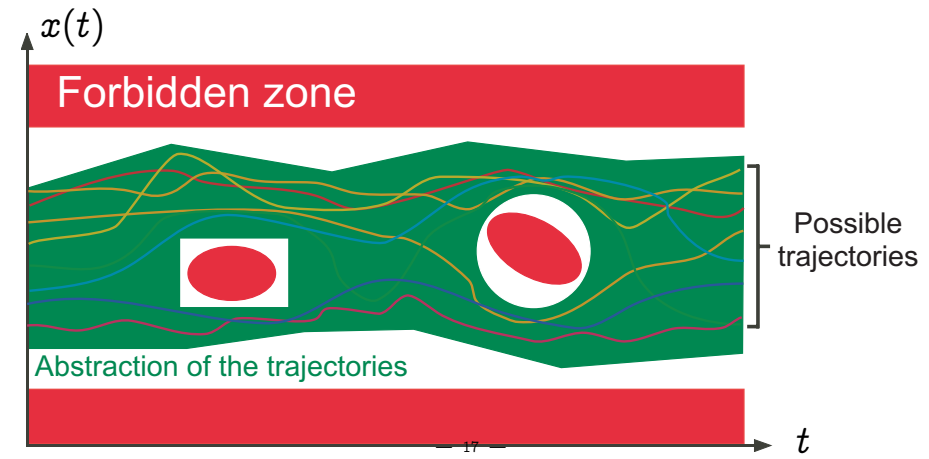
Graphic example: Property test/simulation



Abstract interpretation

- consists in considering an **abstract semantics**, that is to say a superset of the concrete semantics of the program;
- hence the abstract semantics **covers all possible concrete cases**;
- **correct**: if the abstract semantics is safe (does not intersect the forbidden zone) then so is the concrete semantics

Graphic example: Abstract interpretation



Formal methods

Formal methods are abstract interpretations, which differ in the way to obtain the abstract semantics:

- “*model checking*”:
 - the abstract semantics is given manually by the user;
 - in the form of a finitary model of the program execution;
 - can be computed automatically, by techniques relevant to static analysis.

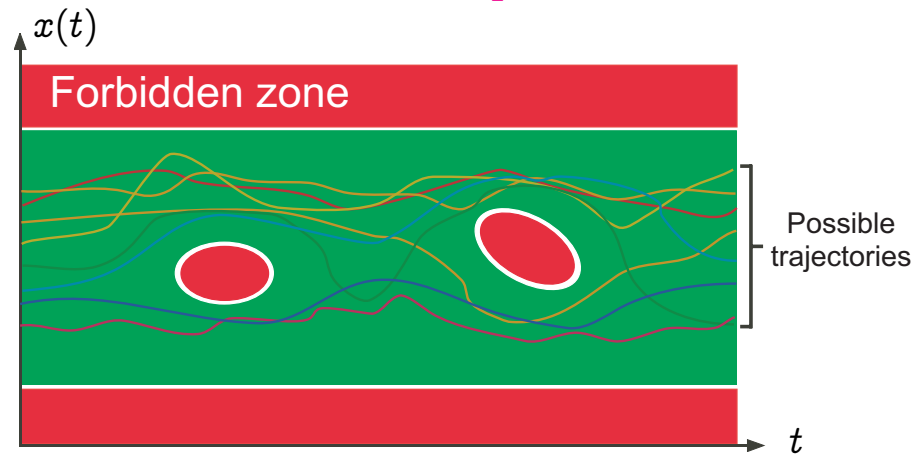
- “*deductive methods*”:
 - the abstract semantics is specified by verification conditions;
 - the user must provide the abstract semantics in the form of inductive arguments (e.g. invariants);
 - can be computed automatically by methods relevant to static analysis.
- “*static analysis*”: the abstract semantics is computed automatically from the program text according to pre-defined abstractions (that can sometimes be tailored automatically/manually by the user).

— 19 —

Required properties of the abstract semantics

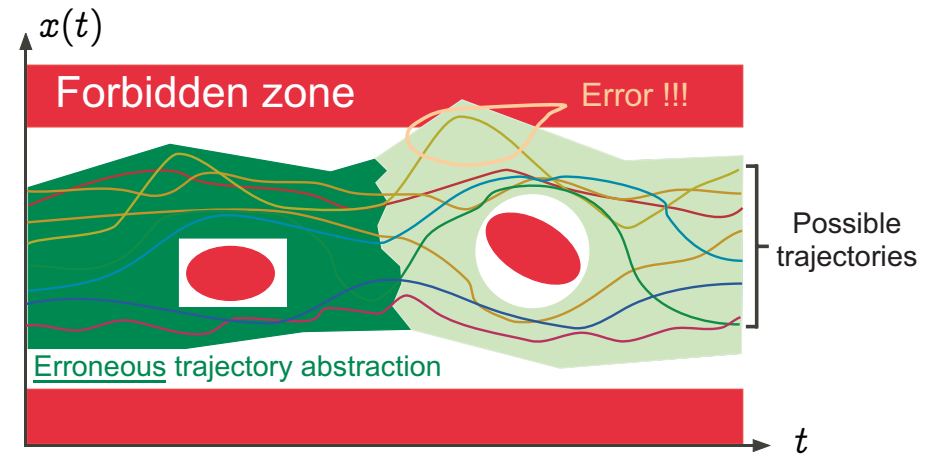
- **sound** so that no possible error can be forgotten;
- **precise** enough (to avoid false alarms);
- as **simple/abstract** as possible (to avoid combinatorial explosion phenomena).

Graphic example: The most abstract correct and precise semantics



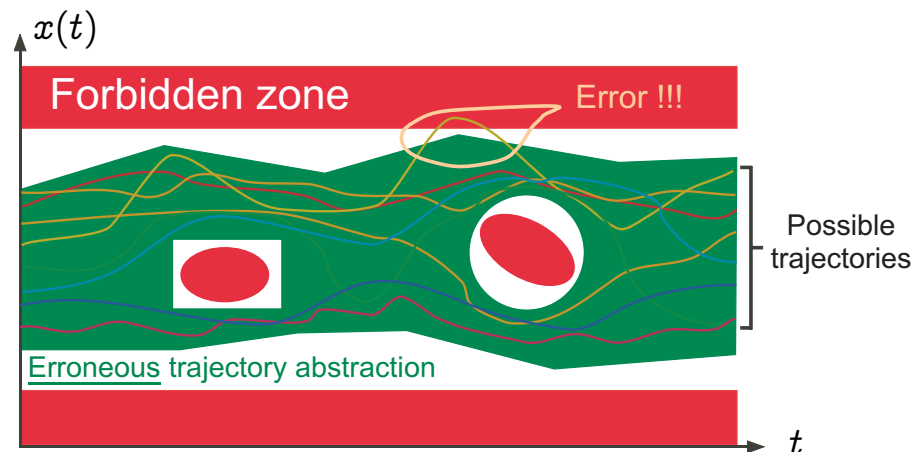
— 21 —

Graphic example: Erroneous abstraction — II

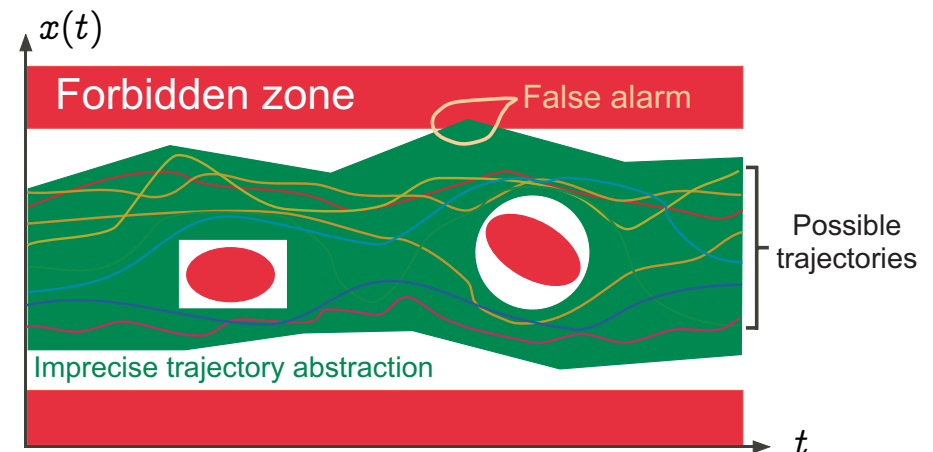


— 23 —

Graphic example: Erroneous abstraction — I



Graphic example: Imprecision \Rightarrow false alarms



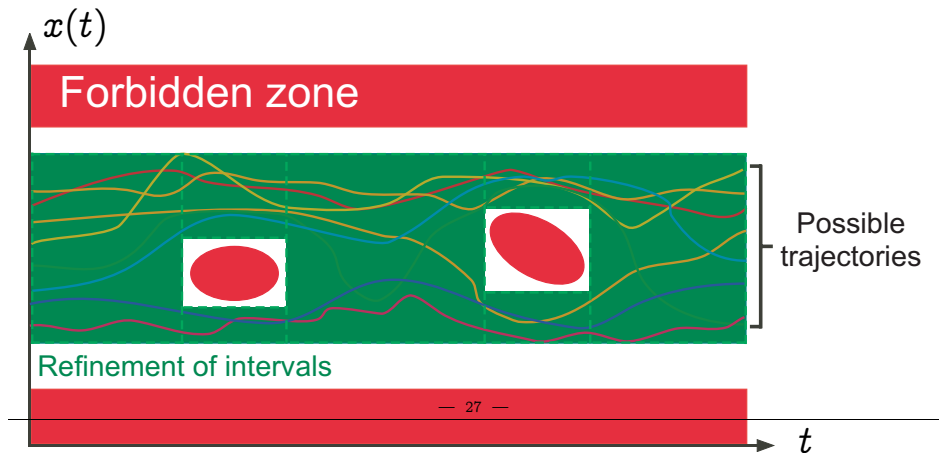
Abstract domains

Standard abstractions

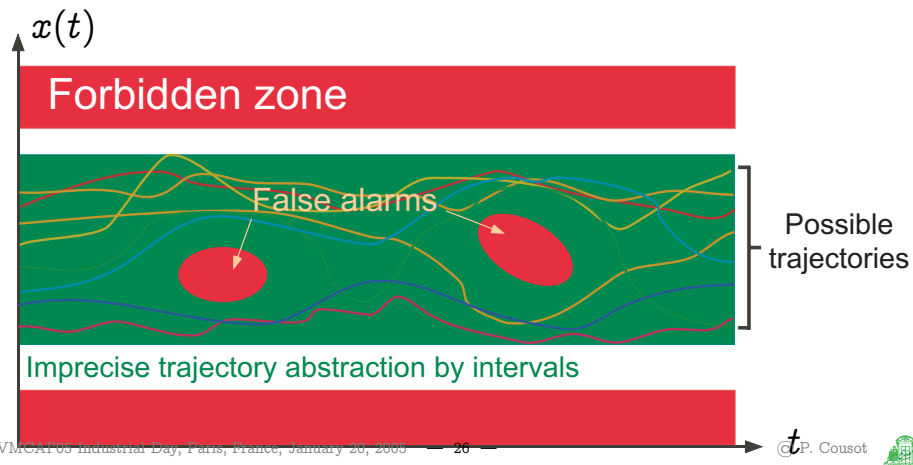
- that serve as a **basis** for the design of static analyzers:
 - abstract program data,
 - abstract program basic operations;
 - abstract program control (iteration, procedure, concurrency, ...);
- can be **parametrized** to allow for manual adaptation to the application domains.

— 25 —

Graphic example: A more refined abstraction



Graphic example: Standard abstraction by intervals



A very informal introduction
to static analysis
algorithms

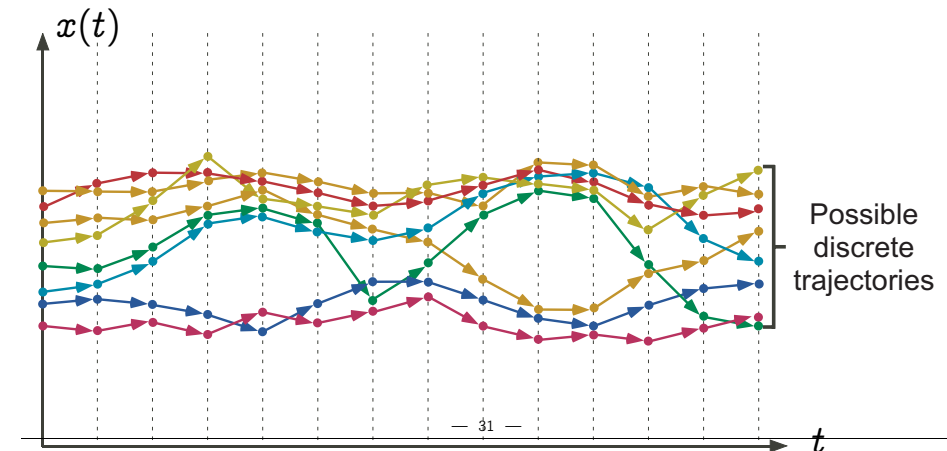
Standard operational semantics

— 29 —

Standard semantics

- Start from a **standard operational semantics** that describes formally:
 - **states** that is data values of program variables,
 - **transitions** that is elementary computation steps;
- Consider **traces** that is successions of states corresponding to executions described by transitions (possibly infinite).

Graphic example: Small-steps transition semantics



Example: Small-steps transition semantics of an assignment

```
int x;
...
l:
    x := x + 1;
l':
```

$$\{l : x = v \rightarrow l' : x = v + 1 \mid v \in [\text{min_int}, \text{max_int} - 1]\} \cup \{l : x = \text{max_int} \rightarrow l' : x = \Omega\} \quad (\text{runtime error})$$

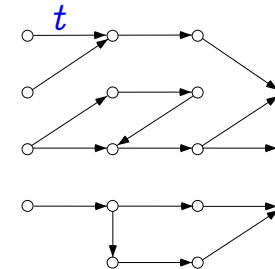
Example: Small-steps transition semantics of a loop

11:	$x := 1;$	11 : ...	$\left. \begin{array}{l} \searrow \\ \rightarrow 12 : x = 1 \\ \nearrow \end{array} \right\}$
12:	while $x < 10$ do	11 : $x = -1$	
13:	$x := x + 1$	11 : $x = 0$	
14:	od	11 : $x = 1$	
15:		11 : ...	
		12 : $x = 1 \rightarrow 13 : x = 1$	
		13 : $x = 1 \rightarrow 14 : x = 2$	
		14 : $x = 2 \rightarrow 13 : x = 2$	
		13 : $x = 2 \rightarrow 14 : x = 3$	
		...	
		14 : $x = 10 \rightarrow 15 : x = 10$	

— 33 —

Transition systems

- $\langle S, \xrightarrow{t} \rangle$ where:
 - S is a set of states/vertices/...
 - $\xrightarrow{t} \in \wp(S \times S)$ is a transition relation/set of arcs/...



— 35 —

Example: Trace semantics of loop

11 : ...	$\left. \begin{array}{l} \searrow \\ \rightarrow 12 : x = 1 \rightarrow 13 : x = 1 \\ \nearrow \end{array} \right\}$	11:	$x := 1;$
11 : $x = -1$		12:	while $x < 10$ do
11 : $x = 0$		13:	$x := x + 1$
11 : $x = 1$		14:	od
11 : ...		15:	
13 : $x = 2 \rightarrow 14 : x = 3 \dots \rightarrow 14 : x = 10$			

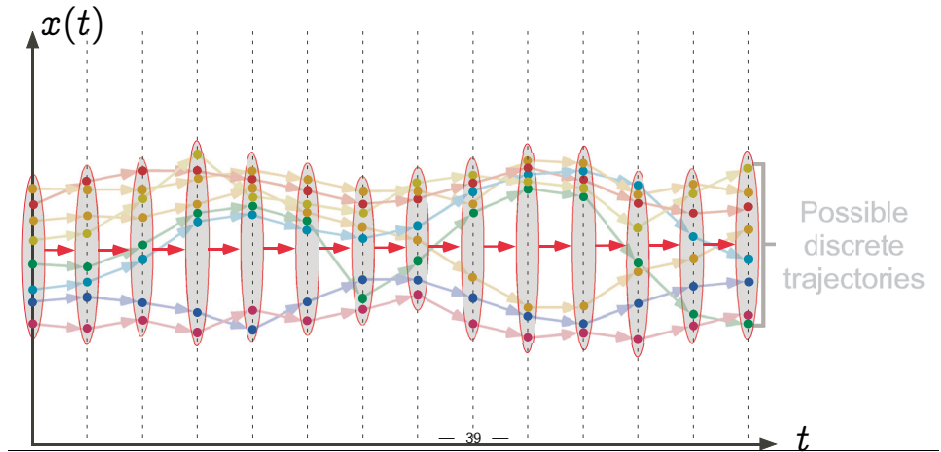
Collecting semantics
in fixpoint form

Collecting semantics

- consider all traces simultaneously;
- collecting semantics:
 - sets of states that describe data values of program variables on all possible trajectories;
 - set of states transitions that is simultaneous elementary computation steps on all possible trajectories;

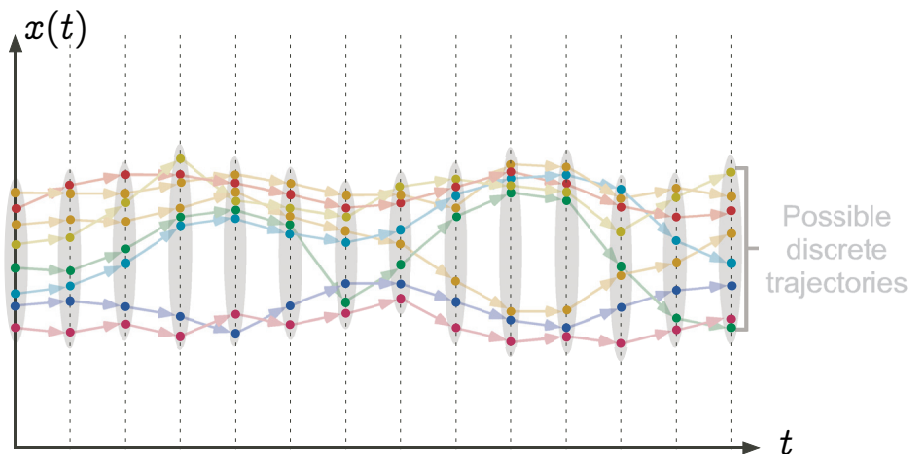
— 37 —

Graphic example: set of states transitions

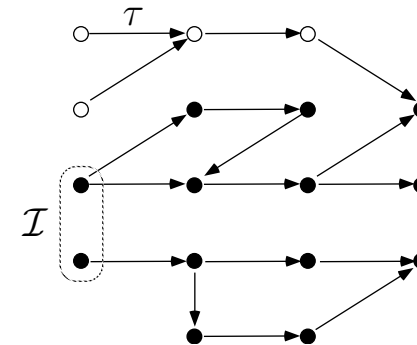


— 39 —

Graphic example: sets of states



Example: Reachable states of a transition system



Reachable states in fixpoint form

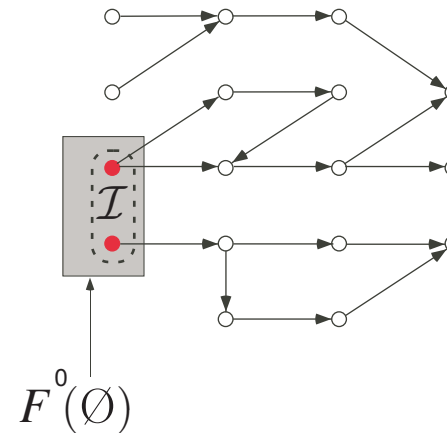
$$F(X) = \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$$

$$\mathcal{R} = \text{lfp}_{\emptyset}^{\subseteq} F$$

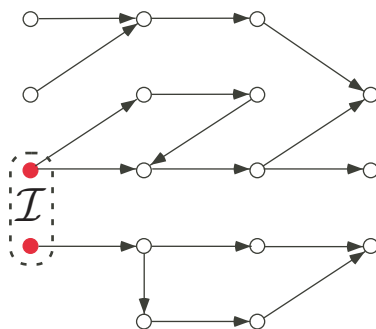
$$= \bigcup_{n=0}^{+\infty} F^n(\emptyset) \quad \text{where} \quad \begin{aligned} f^0(x) &= x \\ f^{n+1}(x) &= f(f^n(x)) \end{aligned}$$

— 41 —

Example of fixpoint iteration
for reachable states $\text{lfp}_{\emptyset}^{\subseteq} \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$

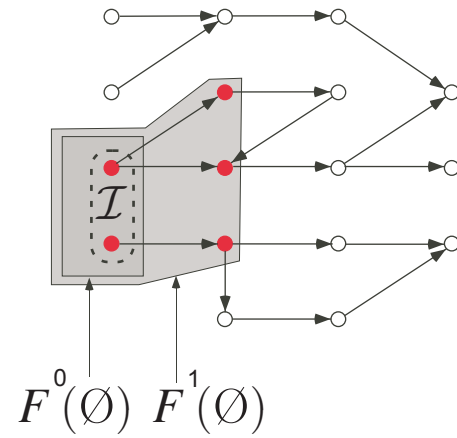


Example of fixpoint iteration
for reachable states $\text{lfp}_{\emptyset}^{\subseteq} \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$

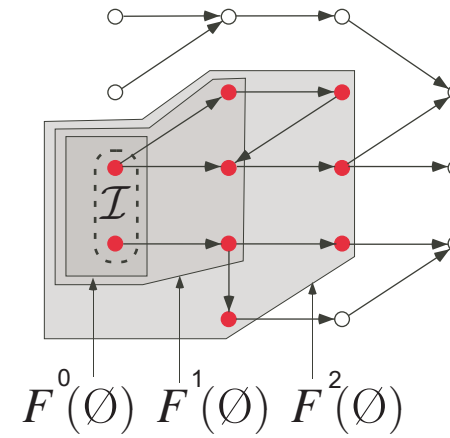


\emptyset

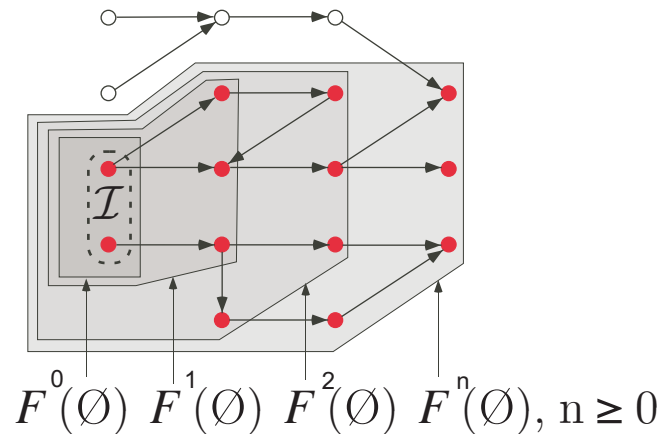
Example of fixpoint iteration
for reachable states $\text{wp}_{\emptyset}^{\subseteq} \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$



Example of fixpoint iteration
for reachable states $\text{wp}_{\emptyset}^{\subseteq} \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$



Example of fixpoint iteration
for reachable states $\text{fp}_\emptyset \subseteq \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$



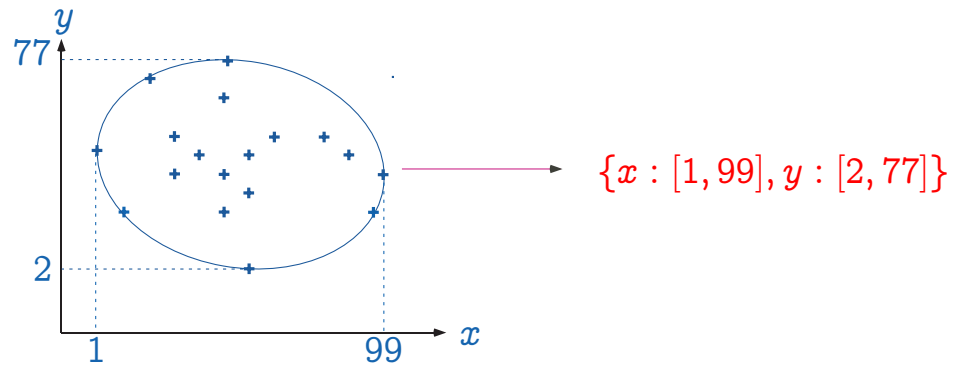
Abstraction by Galois connections

— 43 —

Abstracting sets (i.e. properties)

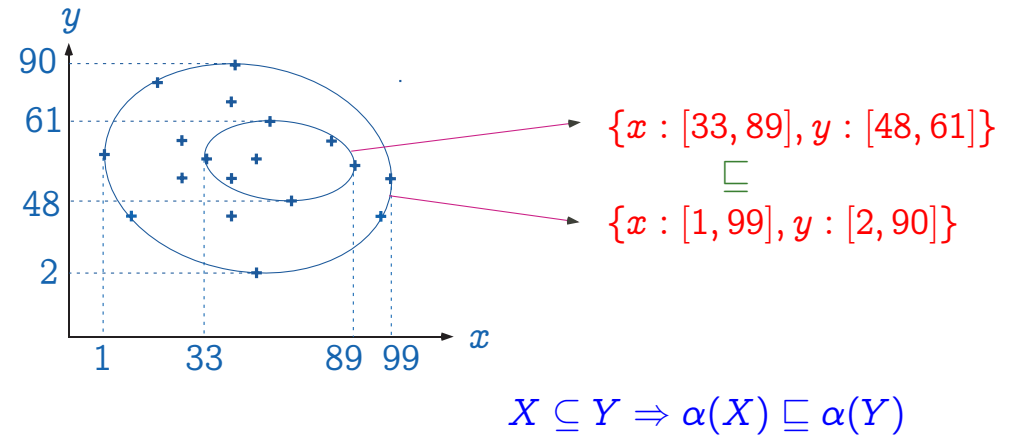
- Choose an **abstract domain**, replacing sets of objects (states, traces, ...) S by their abstraction $\alpha(S)$
- The **abstraction function** α maps a set of concrete objects to its abstract interpretation;
- The inverse **concretization function** γ maps an abstract set of objects to concrete ones;
- **Forget no concrete objects**: (abstraction from above) $S \subseteq \gamma(\alpha(S))$.

Interval abstraction α



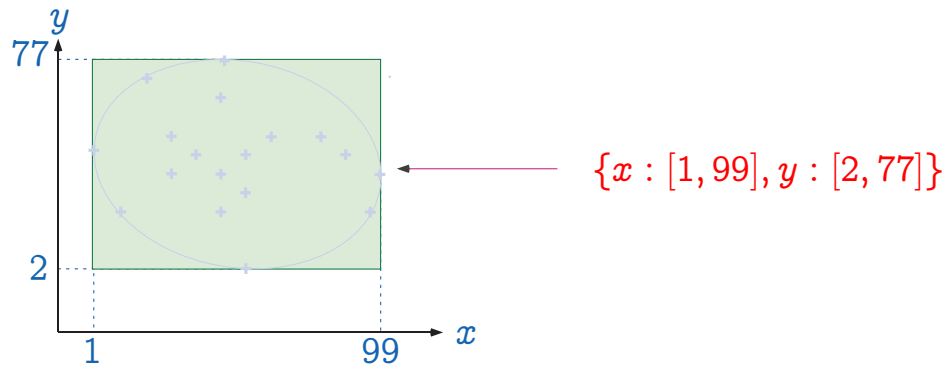
— 45 —

The abstraction α is monotone

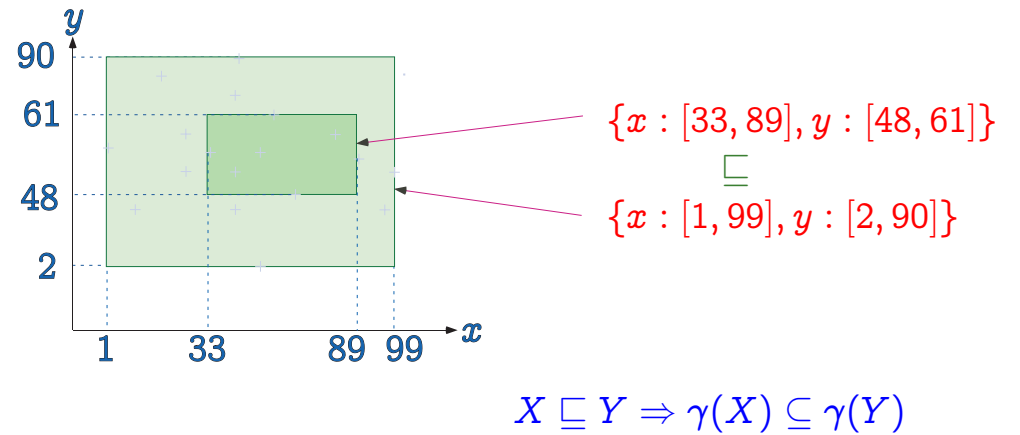


— 47 —

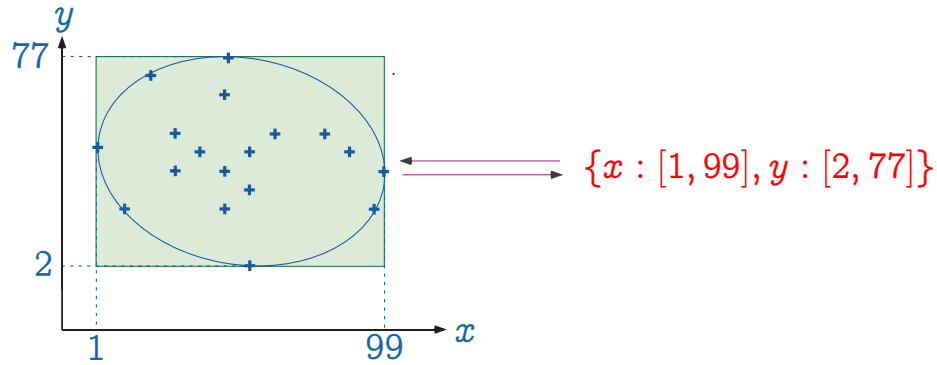
Interval concretization γ



The concretization γ is monotone



The $\gamma \circ \alpha$ composition is extensive



$$X \subseteq \gamma \circ \alpha(X)$$

— 49 —

Correspondance between concrete and abstract properties

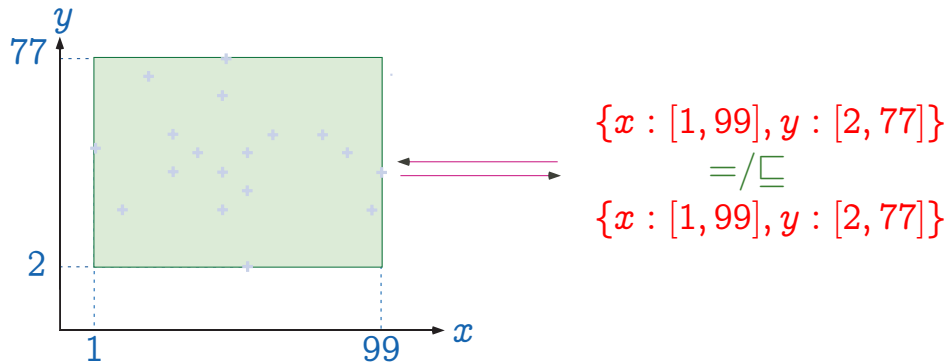
– The pair $\langle \alpha, \gamma \rangle$ is a **Galois connection**:

$$\langle \wp(S), \subseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle \mathcal{D}, \sqsubseteq \rangle$$

– $\langle \wp(S), \subseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle \mathcal{D}, \sqsubseteq \rangle$ when α is onto (equivalently $\alpha \circ \gamma = 1$ or γ is one-to-one).

— 51 —

The $\alpha \circ \gamma$ composition is reductive

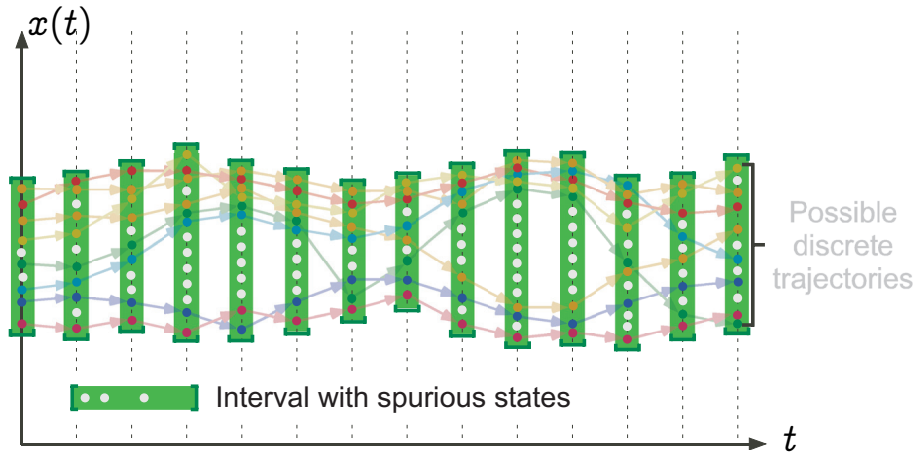


$$\alpha \circ \gamma(Y) =/= Y$$

iff $\forall x, y \in \mathcal{D} : x \subseteq y \implies \alpha(x) \sqsubseteq \alpha(y)$
 $\wedge \forall \bar{x}, \bar{y} \in \overline{\mathcal{D}} : \bar{x} \sqsubseteq \bar{y} \implies \gamma(\bar{x}) \subseteq \gamma(\bar{y})$
 $\wedge \forall x \in \mathcal{D} : x \subseteq \gamma(\alpha(x))$
 $\wedge \forall \bar{y} \in \overline{\mathcal{D}} : \alpha(\gamma(\bar{y})) \sqsubseteq \bar{y}$

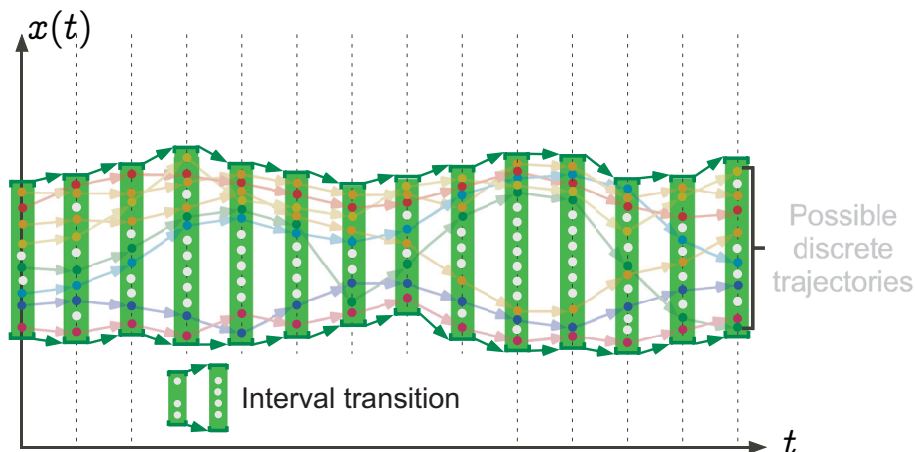
iff $\forall x \in \mathcal{D}, \bar{y} \in \overline{\mathcal{D}} : \alpha(x) \sqsubseteq \bar{y} \iff x \subseteq \gamma(\bar{y})$

Graphic example: Interval abstraction



— 53 —

Graphic example: Abstract transitions



Example: Interval transition semantics of assignments

```
int x;
```

```
...
```

```
l:
```

```
  x := x + 1;
```

```
l':
```

$$\{l : x \in [\ell, h] \rightarrow l' : x \in [\ell + 1, \min(h + 1, \text{max_int})] \cup \{\Omega \mid h = \text{max_int}\} \mid \ell \leq h\}$$

where $[\ell, h] = \emptyset$ when $h < \ell$.

— 55 —

Abstract domain

F^\sharp

Function abstraction

$$F^\sharp = \alpha \circ F \circ \gamma$$

$$\text{i.e. } F^\sharp = \rho \circ F$$

F

Concrete domain

$$\langle P, \sqsubseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle Q, \sqsubseteq \rangle \Rightarrow$$

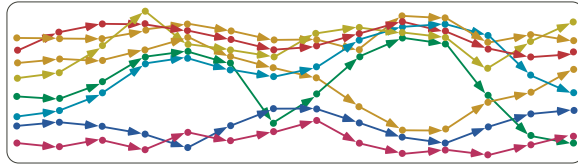
$$\langle P \xrightarrow{\text{mon}} P, \dot{\sqsubseteq} \rangle \xleftrightarrow[\lambda F \cdot \alpha \circ F \circ \gamma]{\lambda F^\sharp \cdot \gamma \circ F^\sharp \circ \alpha} \langle Q \xrightarrow{\text{mon}} Q, \dot{\sqsubseteq} \rangle$$



Example: Set of traces to trace of intervals abstraction

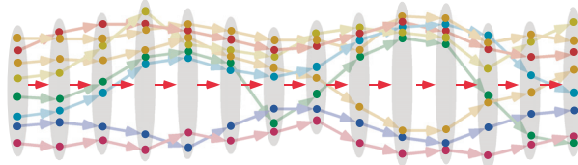
Set of traces:

$\alpha_1 \downarrow$

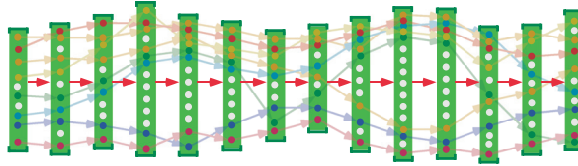


Trace of sets:

$\alpha_2 \downarrow$



Trace of intervals

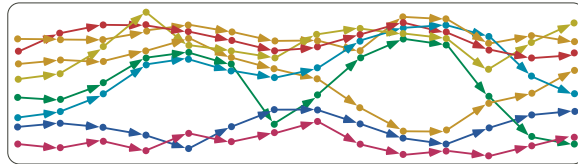


— 57 —

Example: Set of traces to reachable states abstraction

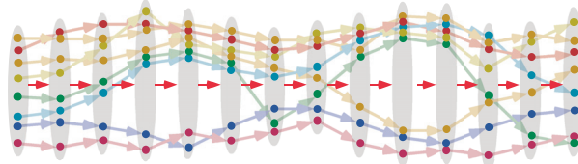
Set of traces:

$\alpha_1 \downarrow$

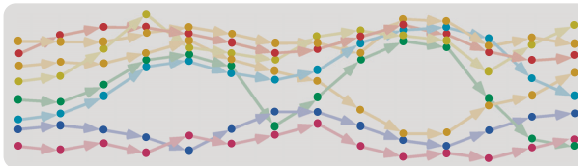


Trace of sets:

$\alpha_3 \downarrow$



Reachable states



Composition of Galois Connections

The composition of Galois connections:

$$\langle L, \leq \rangle \xrightleftharpoons[\alpha_1]{\gamma_1} \langle M, \sqsubseteq \rangle$$

and:

$$\langle M, \sqsubseteq \rangle \xrightleftharpoons[\alpha_2]{\gamma_2} \langle N, \preceq \rangle$$

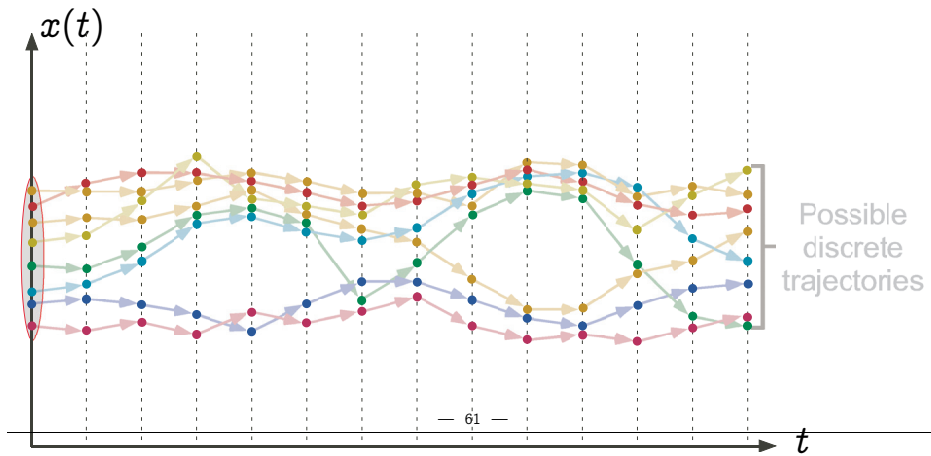
is a Galois connection:

$$\langle L, \leq \rangle \xrightleftharpoons[\alpha_2 \circ \alpha_1]{\gamma_1 \circ \gamma_2} \langle N, \preceq \rangle$$

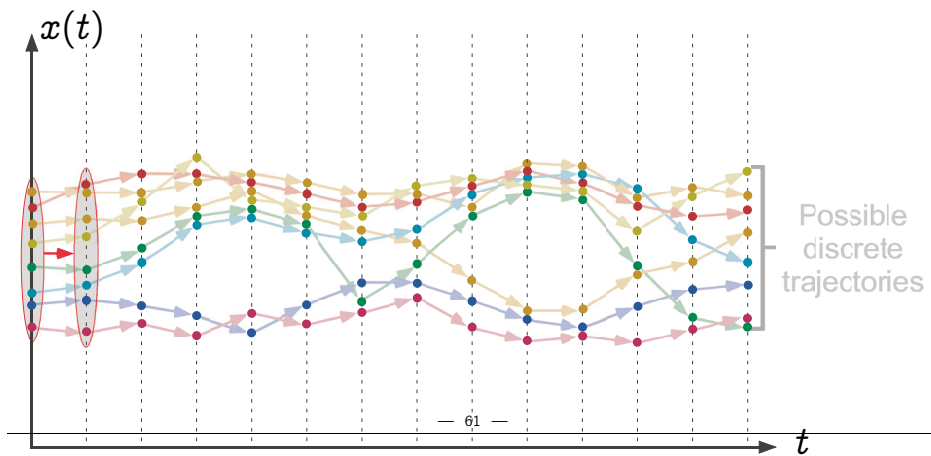
— 59 —

Abstract semantics in fixpoint form

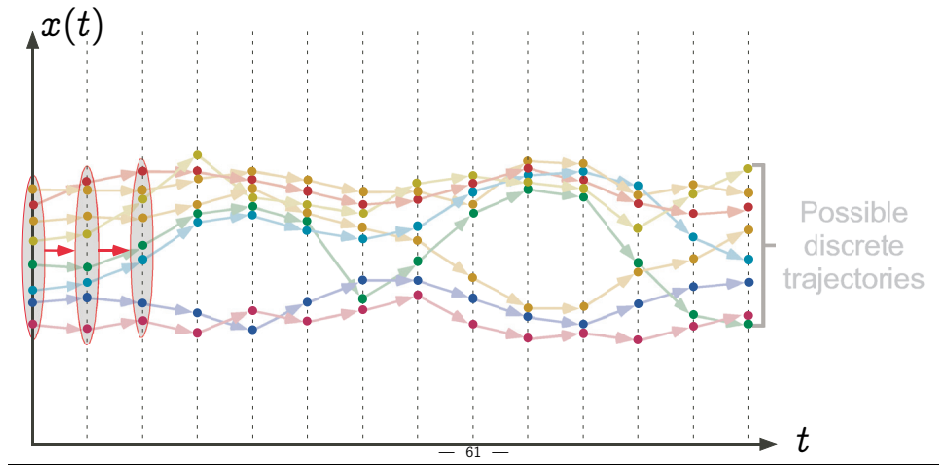
Graphic example: traces of sets of states
in fixpoint form



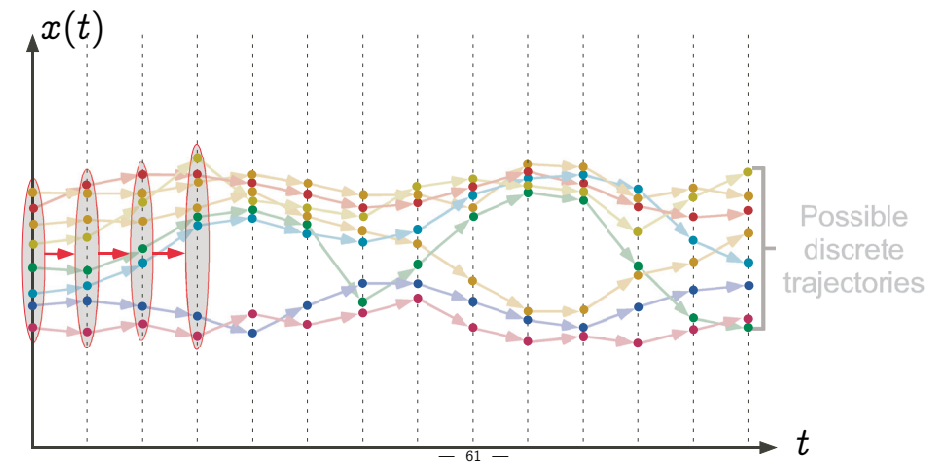
Graphic example: traces of sets of states
in fixpoint form



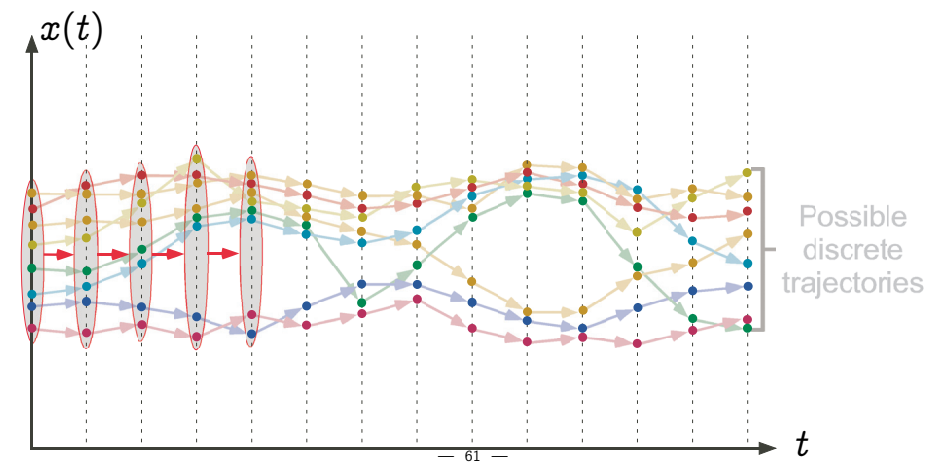
Graphic example: traces of sets of states
in fixpoint form



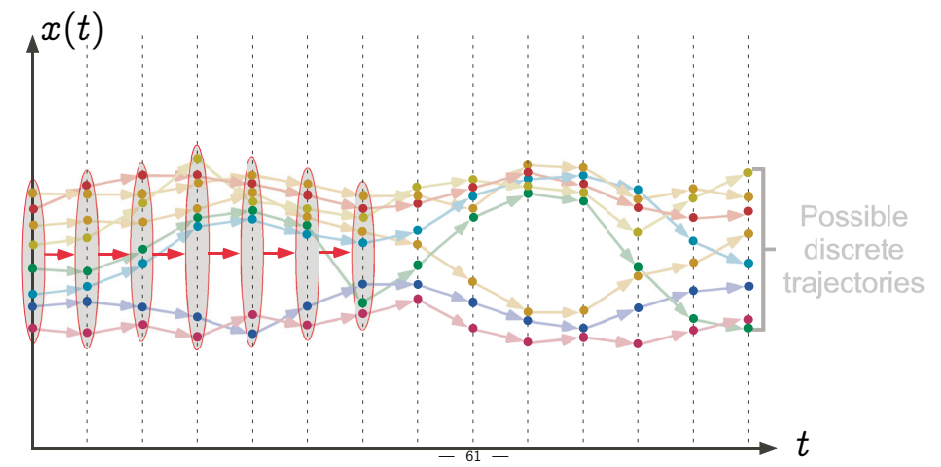
Graphic example: traces of sets of states
in fixpoint form



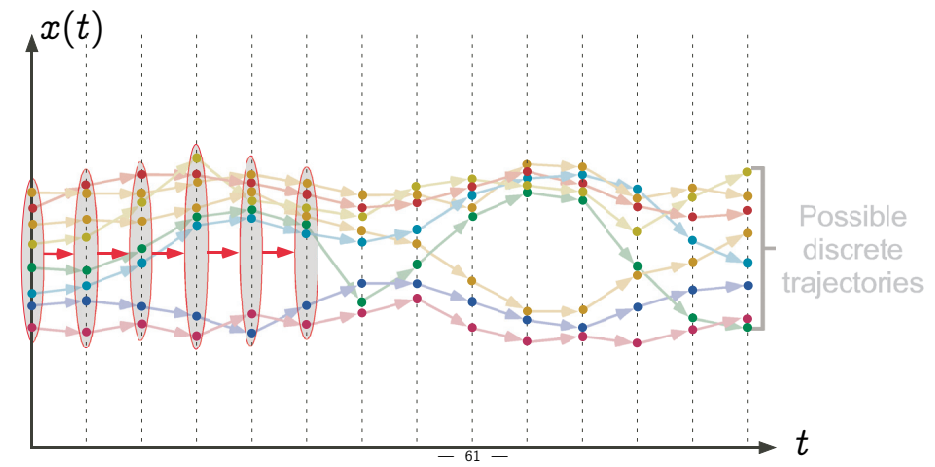
Graphic example: traces of sets of states
in fixpoint form



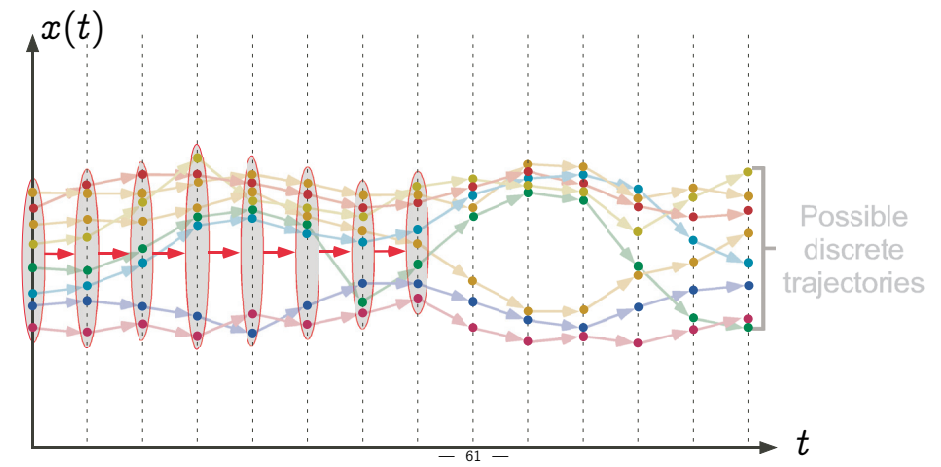
Graphic example: traces of sets of states
in fixpoint form



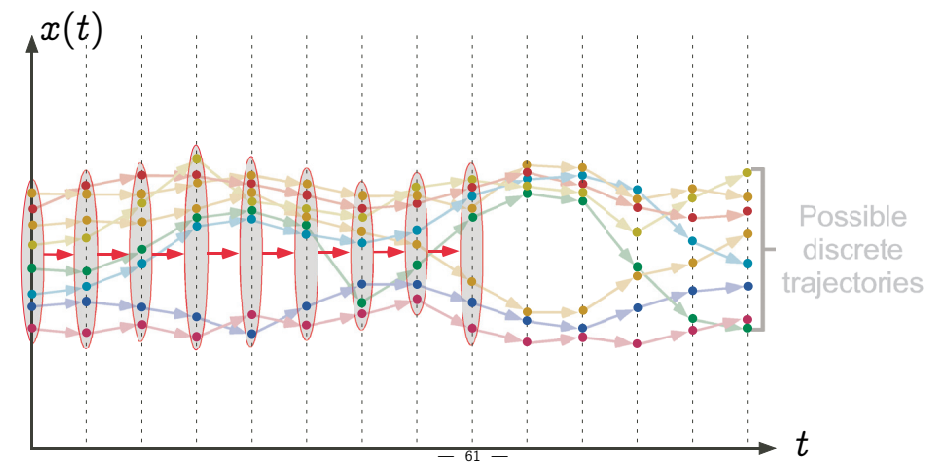
Graphic example: traces of sets of states
in fixpoint form



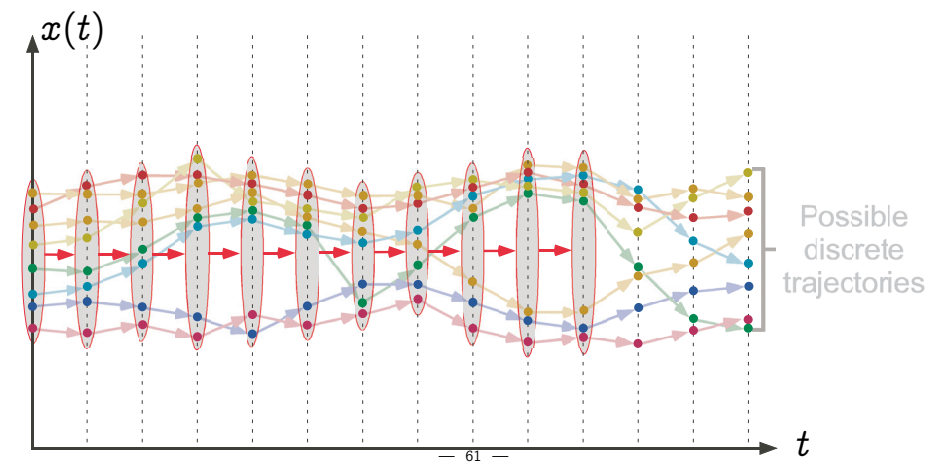
Graphic example: traces of sets of states
in fixpoint form



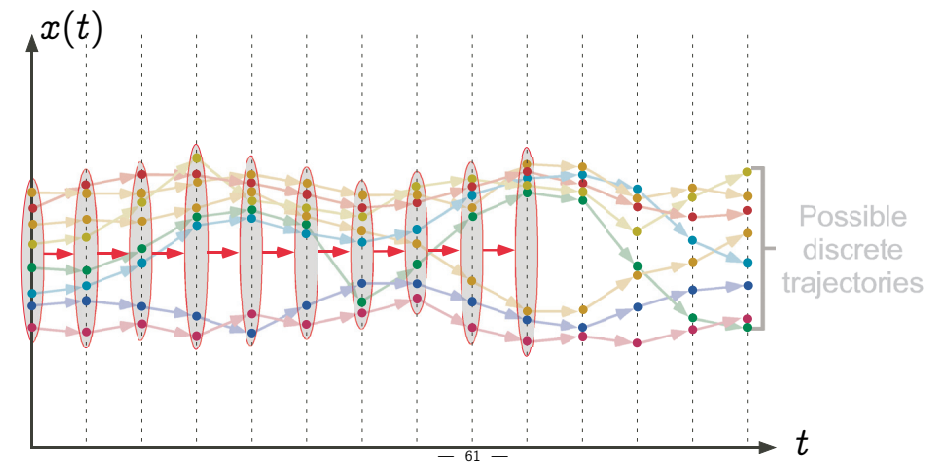
Graphic example: traces of sets of states
in fixpoint form



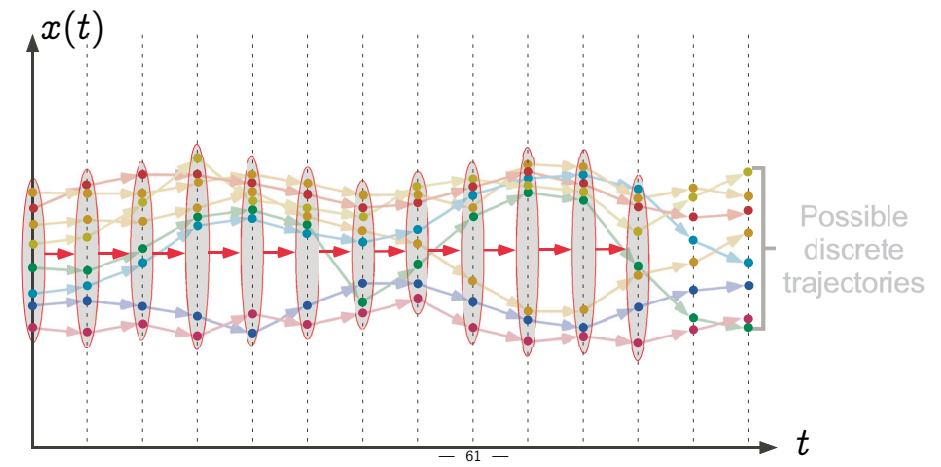
Graphic example: traces of sets of states
in fixpoint form



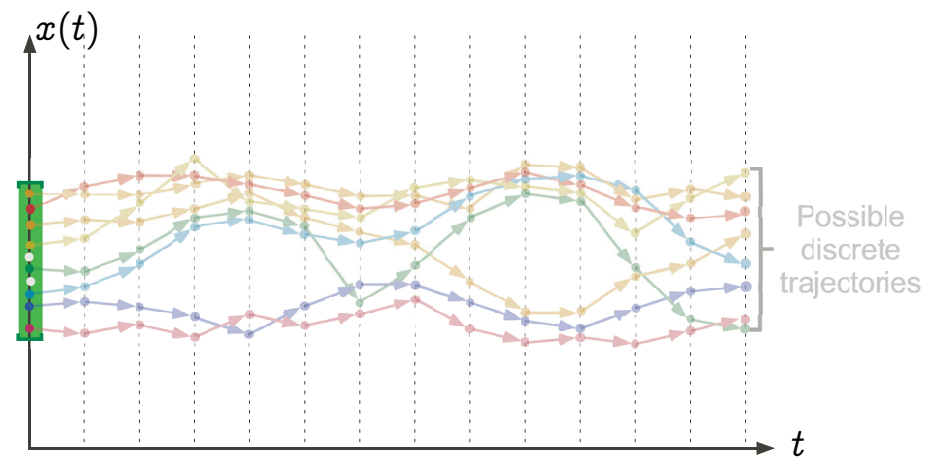
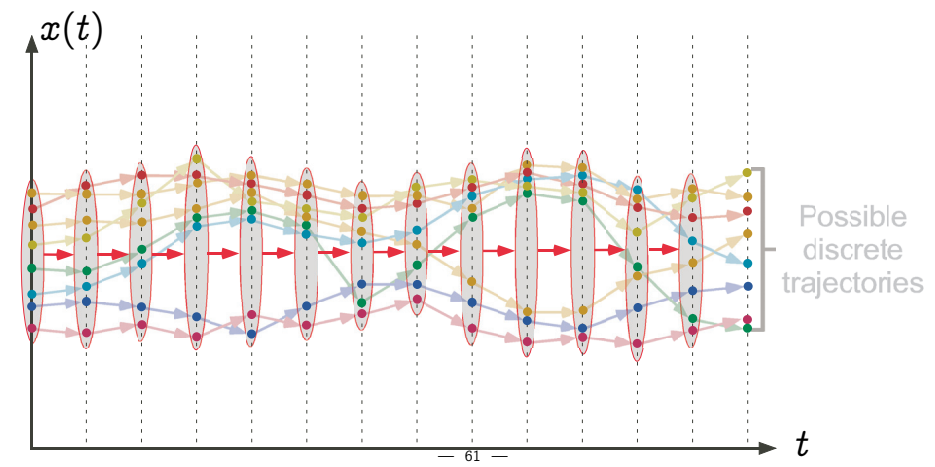
Graphic example: traces of sets of states
in fixpoint form



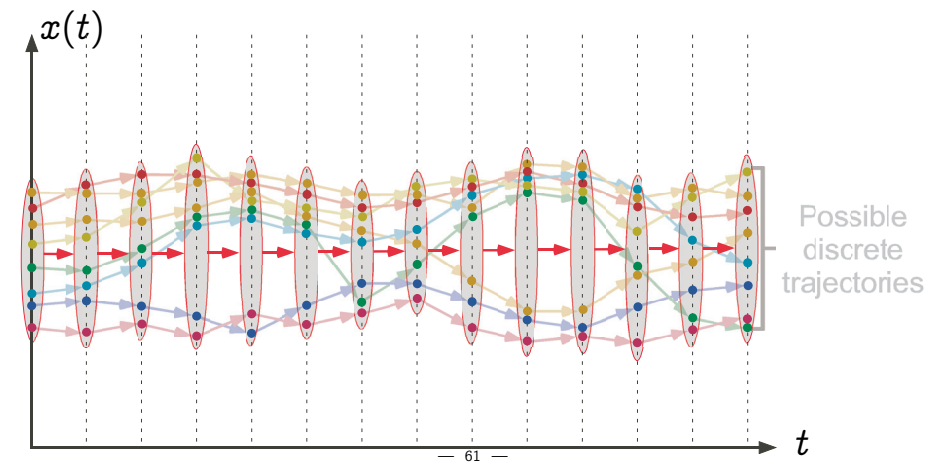
Graphic example: traces of sets of states
in fixpoint form



Graphic example: traces of sets of states
in fixpoint form

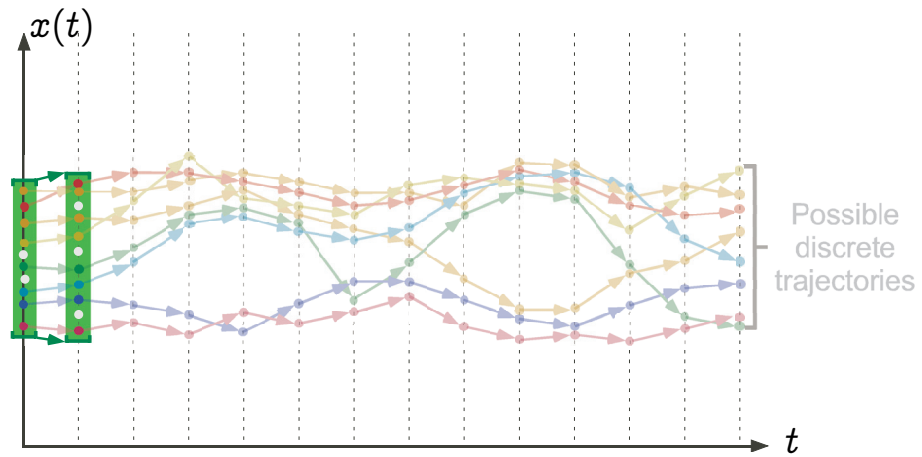


Graphic example: traces of sets of states
in fixpoint form

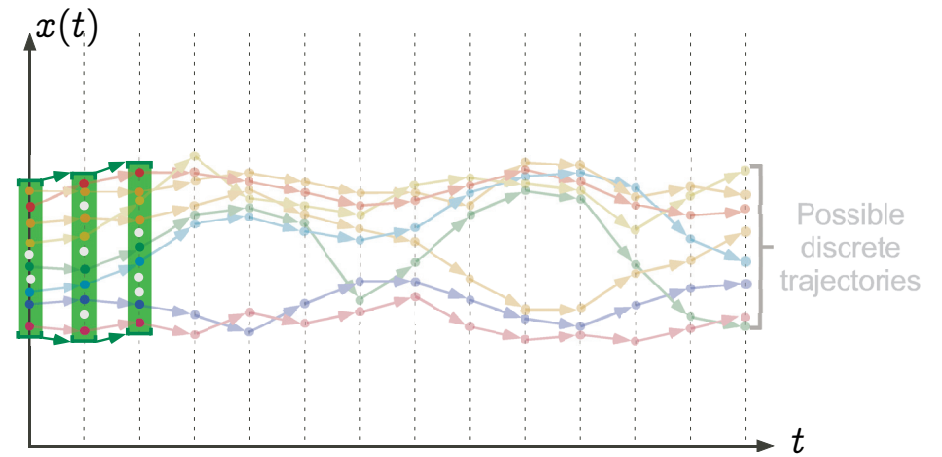


Graphic example: traces of intervals
in fixpoint form

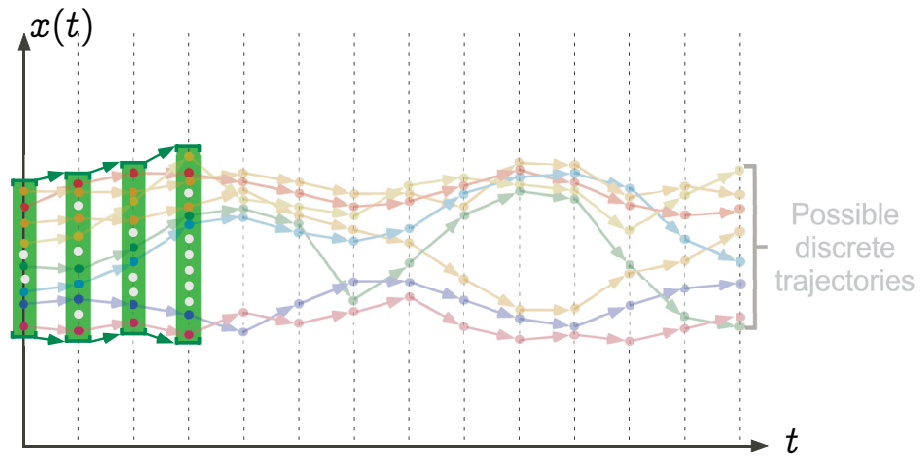
Graphic example: traces of intervals in fixpoint form



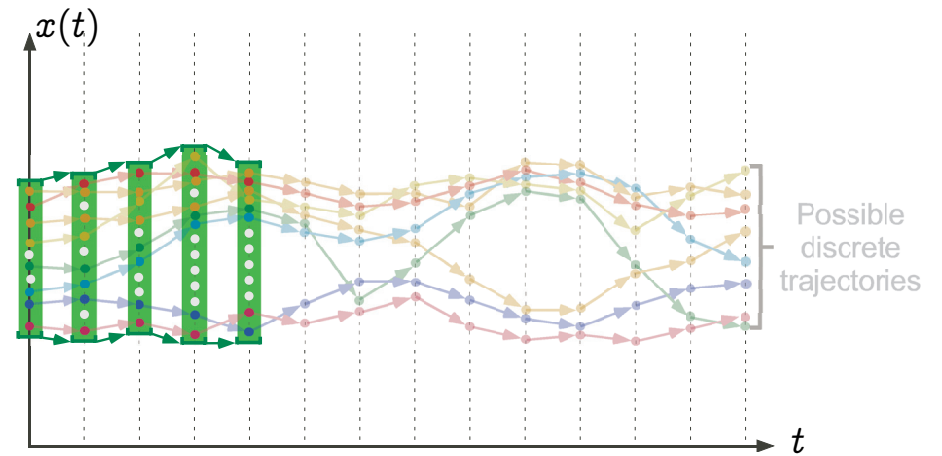
Graphic example: traces of intervals in fixpoint form



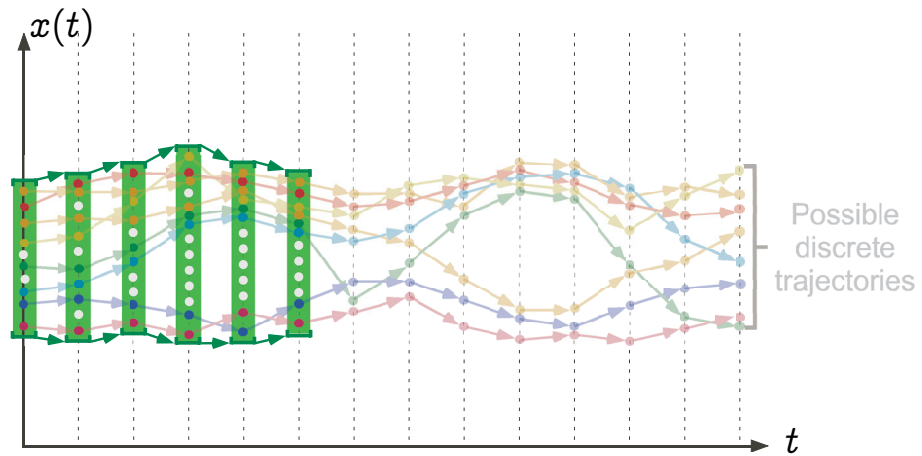
Graphic example: traces of intervals
in fixpoint form



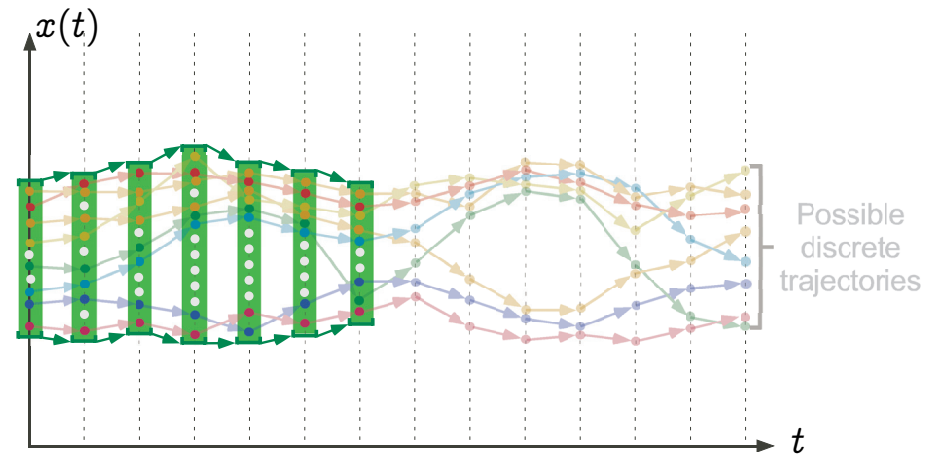
Graphic example: traces of intervals
in fixpoint form



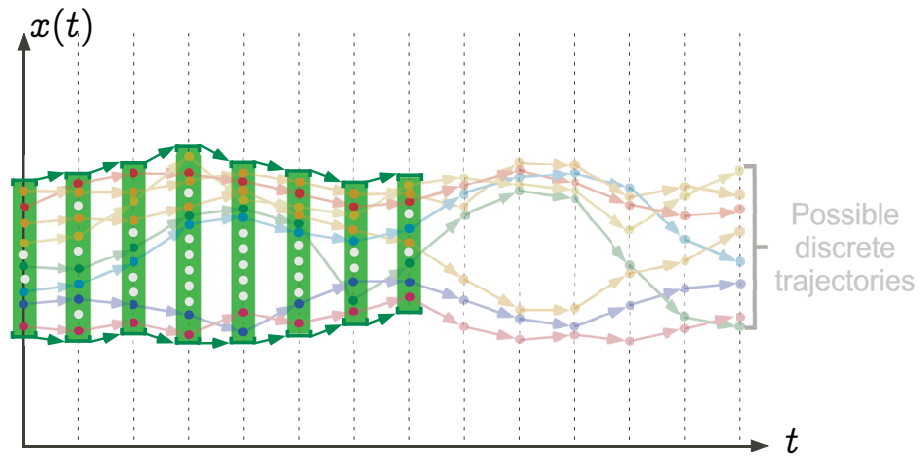
Graphic example: traces of intervals
in fixpoint form



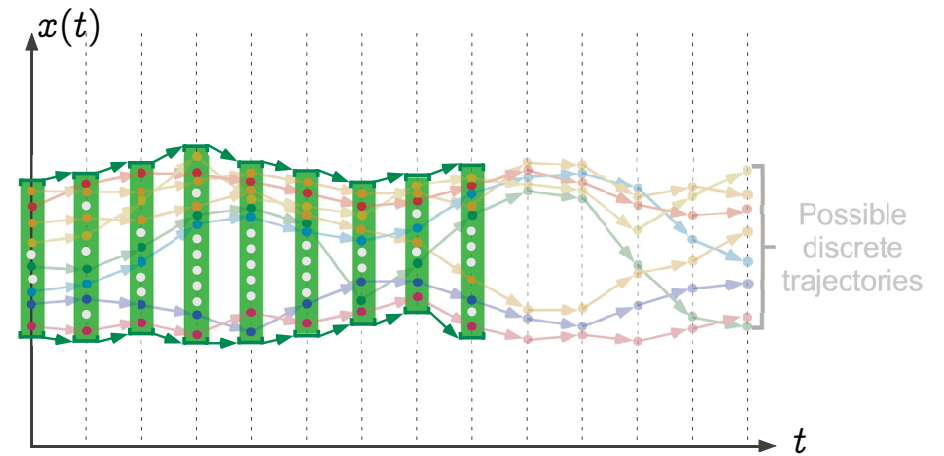
Graphic example: traces of intervals
in fixpoint form



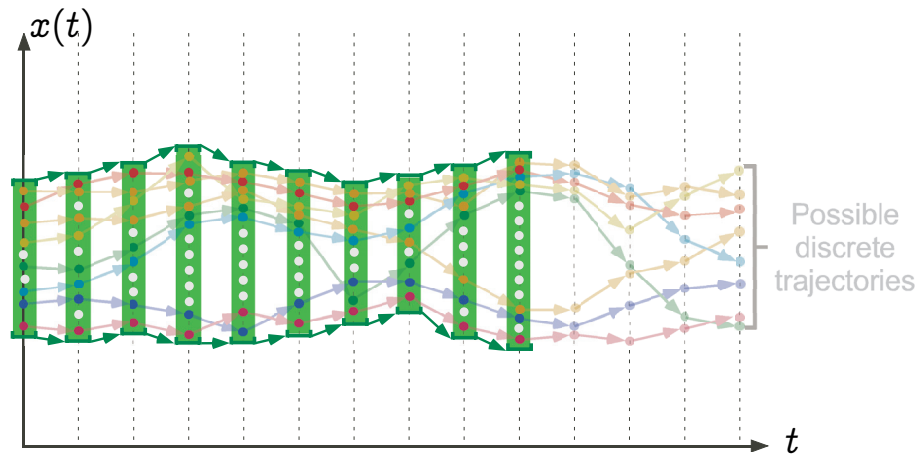
Graphic example: traces of intervals in fixpoint form



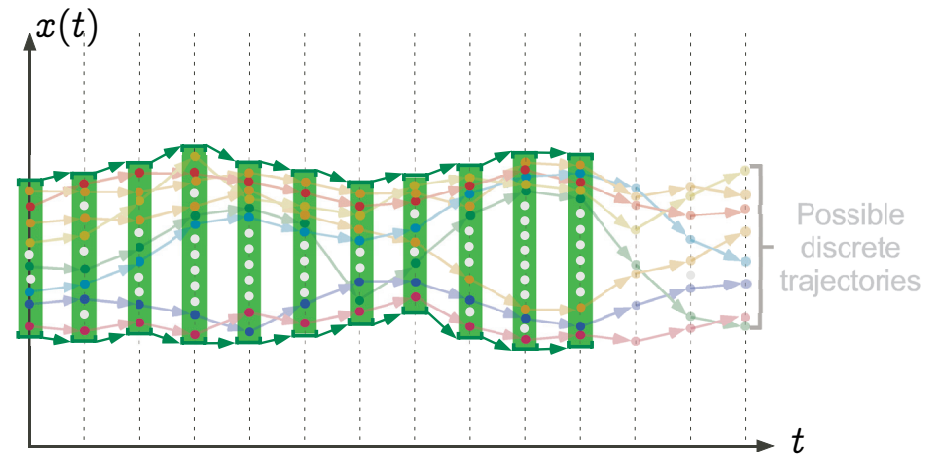
Graphic example: traces of intervals in fixpoint form



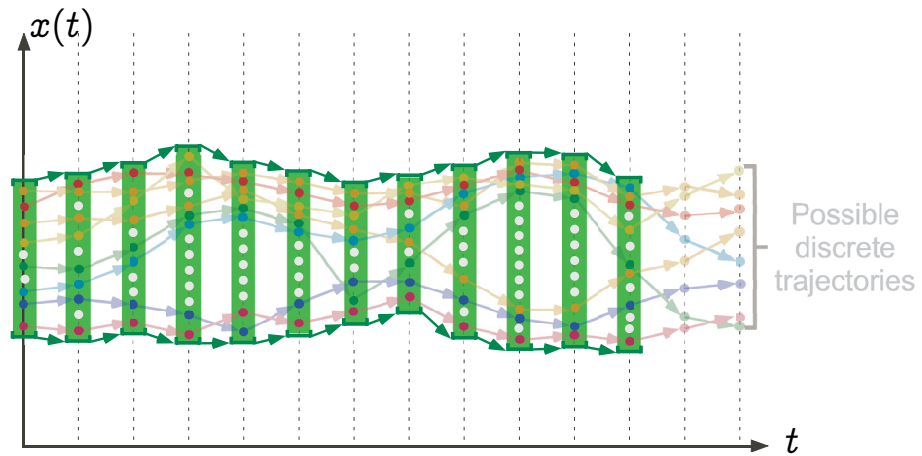
Graphic example: traces of intervals
in fixpoint form



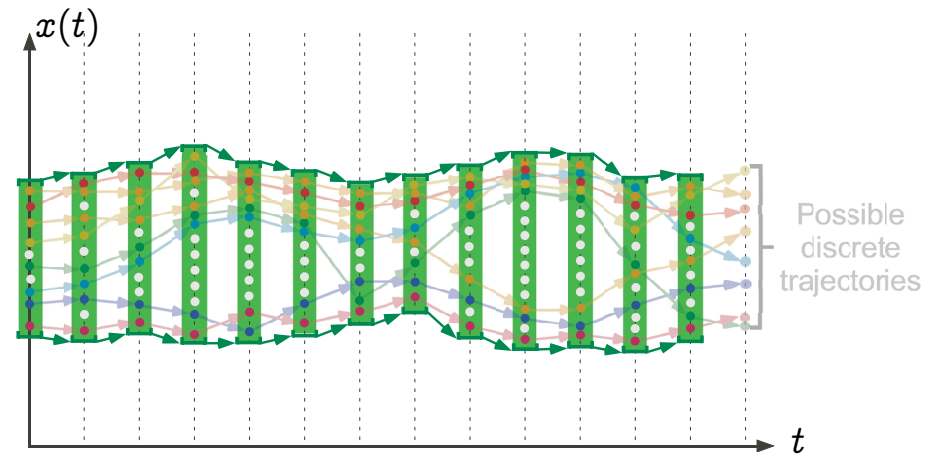
Graphic example: traces of intervals
in fixpoint form



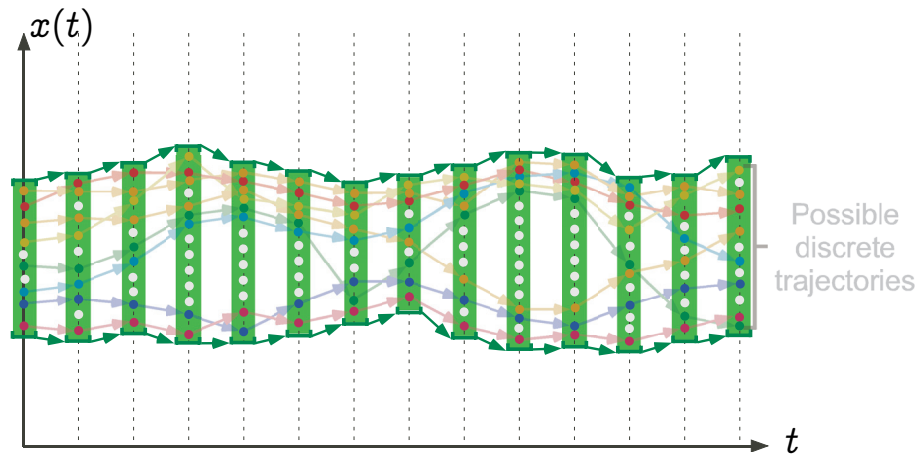
Graphic example: traces of intervals
in fixpoint form



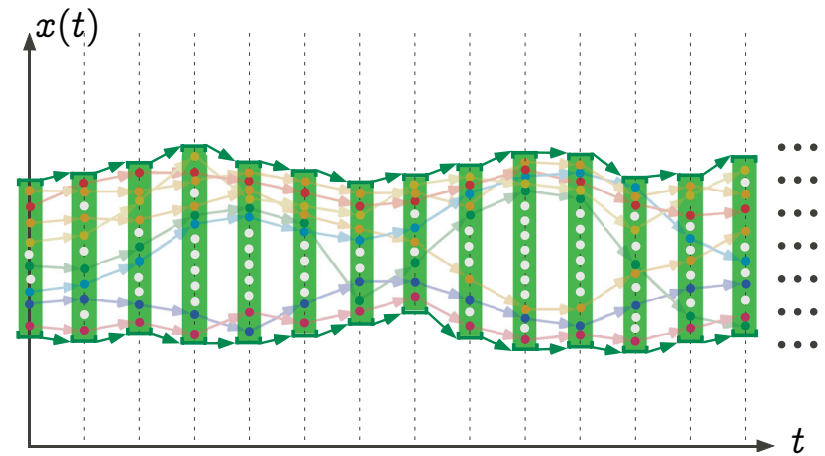
Graphic example: traces of intervals
in fixpoint form



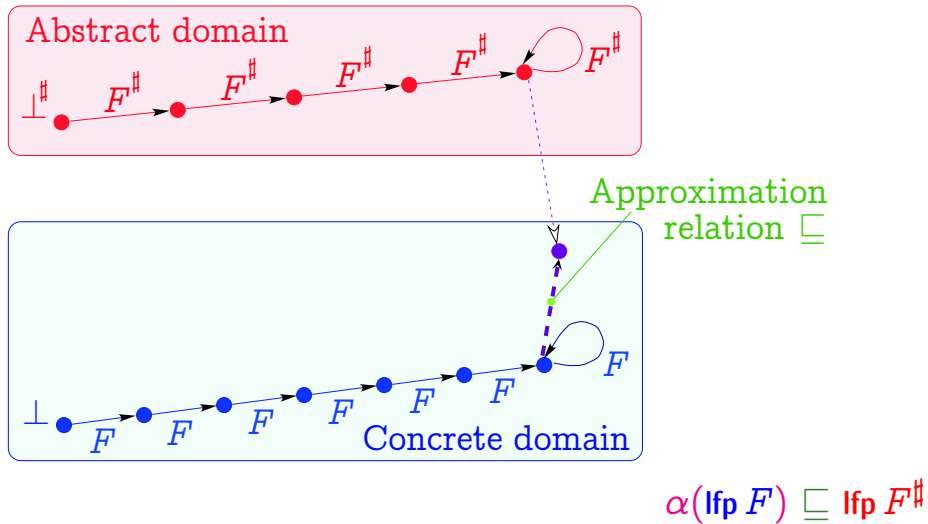
Graphic example: traces of intervals in fixpoint form



Graphic example: traces of intervals in fixpoint form



Approximate fixpoint abstraction



— 63 —

Convergence acceleration
by widening/narrowing

— 65 —

approximate/exact fixpoint abstraction

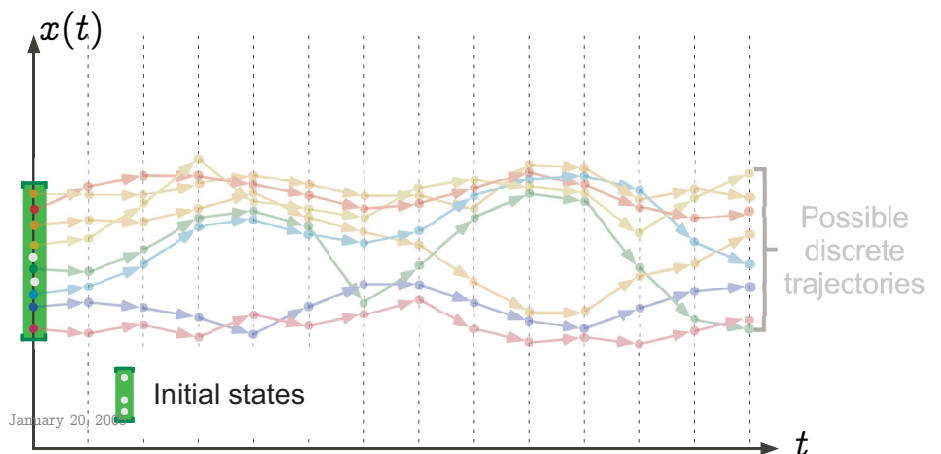
Exact Abstraction:

$$\alpha(\text{lfp } F) = \text{lfp } F^\sharp$$

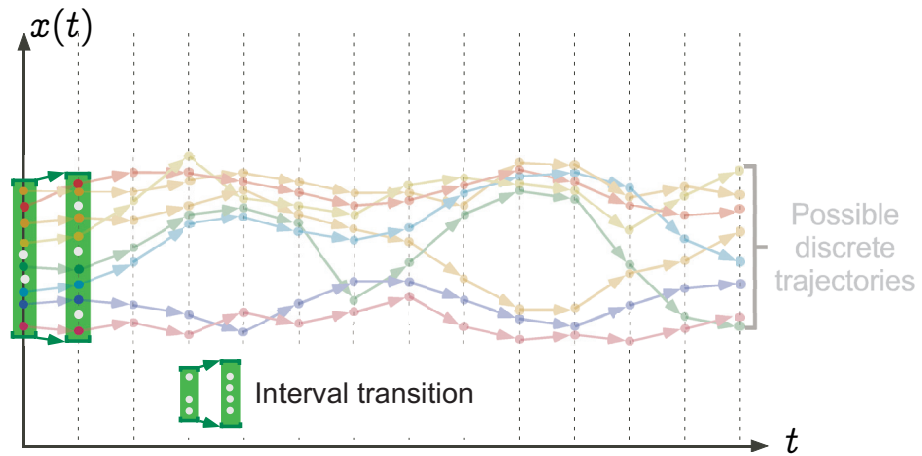
Approximate Abstraction:

$$\alpha(\text{lfp } F) \sqsubseteq^\sharp \text{lfp } F^\sharp$$

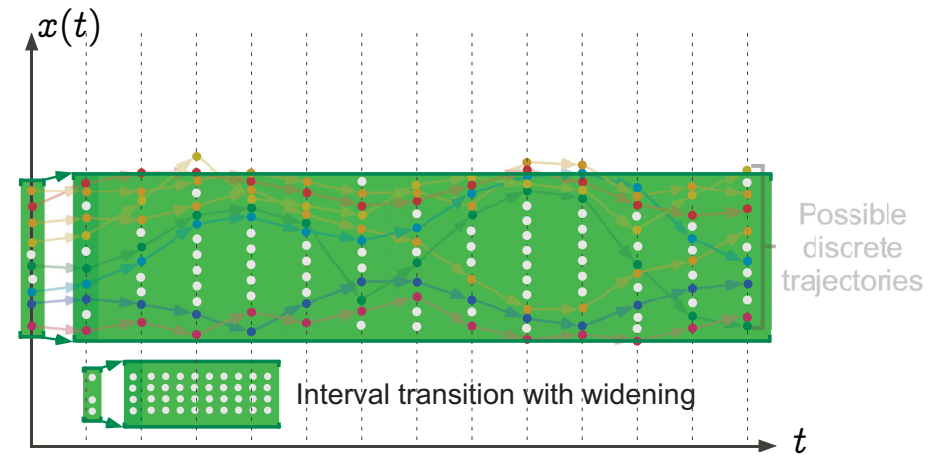
Graphic example: upward iteration with widening



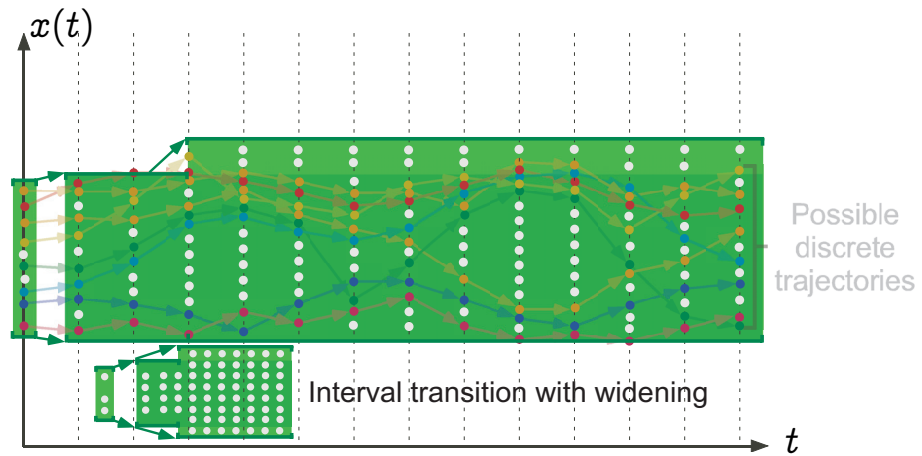
Graphic example: upward iteration with widening



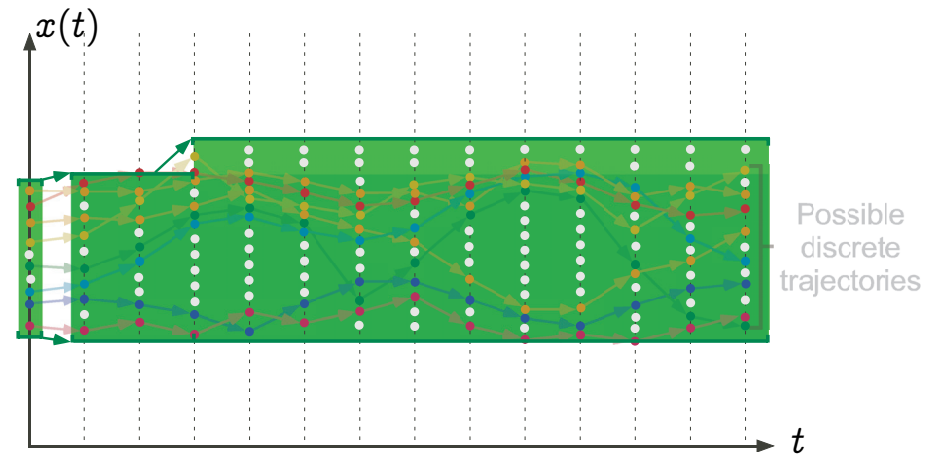
Graphic example: upward iteration with widening



Graphic example: upward iteration with widening



Graphic example: stability of the upward iteration



— 67 —

Convergence acceleration with widening

Widening operator

A widening operator $\nabla \in \overline{L} \times \overline{L} \mapsto \overline{L}$ is such that:

- **Correctness:**
 - $\forall x, y \in \overline{L} : \gamma(x) \sqsubseteq \gamma(x \nabla y)$
 - $\forall x, y \in \overline{L} : \gamma(y) \sqsubseteq \gamma(x \nabla y)$
- **Convergence:**
 - for all increasing chains $x^0 \sqsubseteq x^1 \sqsubseteq \dots$, the increasing chain defined by $y^0 = x^0, \dots, y^{i+1} = y^i \nabla x^{i+1}, \dots$ is not strictly increasing.

— 69 —

Fixpoint approximation with widening

The upward iteration sequence with widening:

- $\hat{X}^0 = \overline{\perp}$ (infimum)
- $\hat{X}^{i+1} = \hat{X}^i$ if $\overline{F}(\hat{X}^i) \sqsubseteq \hat{X}^i$
 $= \hat{X}^i \nabla F(\hat{X}^i)$ otherwise

is ultimately stationary and its limit \hat{A} is a sound upper approximation of $\text{lfp}_{\overline{\perp}} \overline{F}$:

$$\text{lfp}_{\overline{\perp}} \overline{F} \sqsubseteq \hat{A}$$

Interval widening

- $\overline{L} = \{\perp\} \cup \{[\ell, u] \mid \ell, u \in \mathbb{Z} \cup \{-\infty\} \wedge u \in \mathbb{Z} \cup \{\infty\} \wedge \ell \leq u\}$
- The **widening** extrapolates unstable bounds to infinity:

$$\begin{aligned} \perp \nabla X &= X \\ X \nabla \perp &= X \\ [\ell_0, u_0] \nabla [\ell_1, u_1] &= [\text{if } \ell_1 < \ell_0 \text{ then } -\infty \text{ else } \ell_0, \\ &\quad \text{if } u_1 > u_0 \text{ then } +\infty \text{ else } u_0] \end{aligned}$$

Not monotone. For example $[0, 1] \sqsubseteq [0, 2]$ but $[0, 1] \nabla [0, 2] = [0, +\infty] \not\sqsubseteq [0, 2] = [0, 2] \nabla [0, 2]$

— 71 —

Example: Interval analysis (1975)

Program to be analyzed:

```

x := 1;
1:
  while x < 10000 do
2:
    x := x + 1
3:
  od;
4:

```

Example: Interval analysis (1975)

Equations (abstract interpretation of the semantics):

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3: od;
4:

```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

— 73 —

Example: Interval analysis (1975)

Resolution by chaotic increasing iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3: od;
4:

```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = \emptyset \\ X_2 = \emptyset \\ X_3 = \emptyset \\ X_4 = \emptyset \end{cases}$$

Example: Interval analysis (1975)

Increasing chaotic iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3: od;
4:

```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = \emptyset \\ X_3 = \emptyset \\ X_4 = \emptyset \end{cases}$$

— 75 —

Example: Interval analysis (1975)

Increasing chaotic iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3: od;
4:

```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, 1] \\ X_3 = \emptyset \\ X_4 = \emptyset \end{cases}$$

— 76 —

Example: Interval analysis (1975)

Increasing chaotic iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$


```

Example: Interval analysis (1975)

Increasing chaotic iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$


```

Example: Interval analysis (1975)

Increasing chaotic iteration: **convergence !**

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$


```

Example: Interval analysis (1975)

Increasing chaotic iteration: **convergence !!**

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$


```

Example: Interval analysis (1975)

Increasing chaotic iteration: **convergence !!!**

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }

```

Example: Interval analysis (1975)

Increasing chaotic iteration: **convergence !!!!**

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }

```

Example: Interval analysis (1975)

Increasing chaotic iteration: **convergence !!!!!**

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }

```

Example: Interval analysis (1975)

Increasing chaotic iteration: **convergence !!!!!**

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }

```

Example: Interval analysis (1975)

Increasing chaotic iteration: **convergence !!!!!!!**

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   od;
4:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }

```

Example: Interval analysis (1975)

Convergence speed-up by widening:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }
4:   {
      X1 = [1, 1]
      X2 = [1, +∞] ← widening
      X3 = [2, 6]
      X4 = ∅
    }

```

Example: Interval analysis (1975)

Decreasing chaotic iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }
4:   {
      X1 = [1, 1]
      X2 = [1, +∞]
      X3 = [2, +∞]
      X4 = ∅
    }

```

Example: Interval analysis (1975)

Decreasing chaotic iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3:   {
      X1 = [1, 1]
      X2 = (X1 ∪ X3) ∩ [−∞, 9999]
      X3 = X2 ⊕ [1, 1]
      X4 = (X1 ∪ X3) ∩ [10000, +∞]
    }
4:   {
      X1 = [1, 1]
      X2 = [1, 9999]
      X3 = [2, +∞]
      X4 = ∅
    }

```

Example: Interval analysis (1975)

Decreasing chaotic iteration:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3: od;
4:

```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, 9999] \\ X_3 = [2, +10000] \\ X_4 = \emptyset \end{cases}$$

Example: Interval analysis (1975)

Final solution:

```

x := 1;
1: while x < 10000 do
2:   x := x + 1
3: od;
4:

```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, 9999] \\ X_3 = [2, +10000] \\ X_4 = [+10000, +10000] \end{cases}$$

Example: Interval analysis (1975)

Result of the interval analysis:

```

x := 1;
1: {x = 1}
   while x < 10000 do
2: {x ∈ [1, 9999]}
   x := x + 1
3: {x ∈ [2, +10000]}
   od;
4: {x = 10000}

```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, 9999] \\ X_3 = [2, +10000] \\ X_4 = [+10000, +10000] \end{cases}$$

Example: Interval analysis (1975)

Checking absence of runtime errors with interval analysis:

```

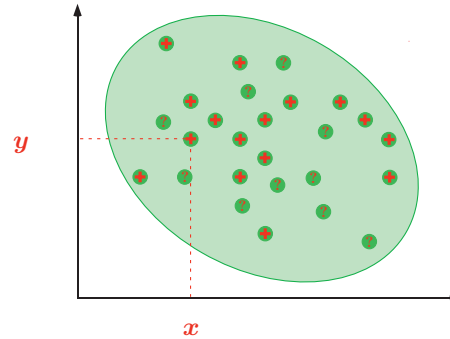
x := 1;
1: {x = 1}
   while x < 10000 do
2: {x ∈ [1, 9999]}
   x := x + 1
3: {x ∈ [2, +10000]}
   od;
4: {x = 10000}

```

← no overflow

Refinement of abstractions

Approximations of an [in]finite set of points:
from above

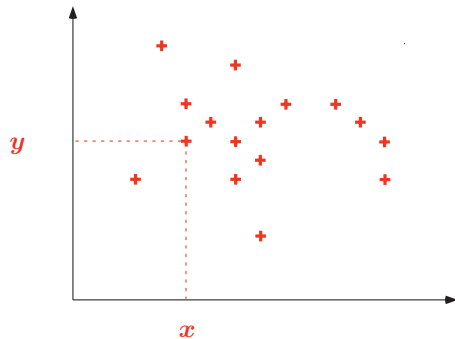


$\{\dots, \langle 19, 77 \rangle, \dots,$
 $\langle 20, 03 \rangle, \langle ?, ? \rangle, \dots\}$

From Below: dual² + combinations.

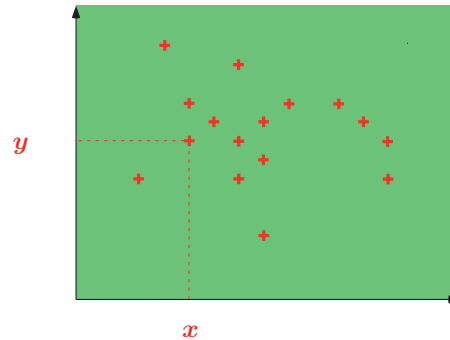
— 83 —

Approximations of an [in]finite set of points:



$\{\dots, \langle 19, 77 \rangle, \dots,$
 $\langle 20, 03 \rangle, \dots\}$

Effective computable approximations of an
[in]finite set of points; Signs³

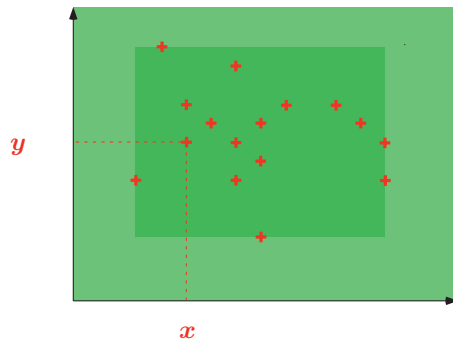


$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$

² Trivial for finite states (liveness model-checking), more difficult for infinite states (variant functions).

³ P. Cousot & R. Cousot. *Systematic design of program analysis frameworks*. ACM POPL'79, pp. 269–282, 1979.

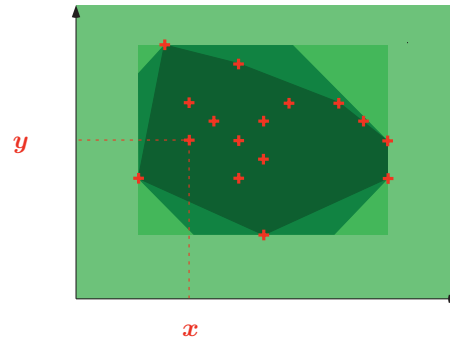
Effective computable approximations of an [in]finite set of points; Intervals⁴



$$\begin{cases} x \in [19, 77] \\ y \in [20, 03] \end{cases}$$

— 85 —

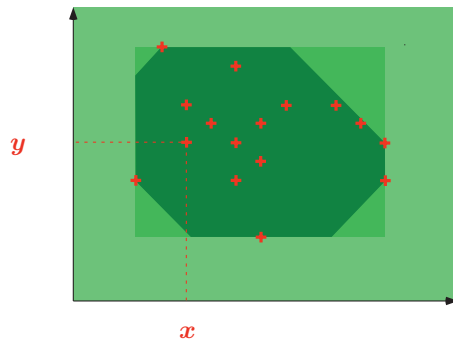
Effective computable approximations of an [in]finite set of points; Polyhedra⁶



$$\begin{cases} 19x + 77y \leq 2004 \\ 20x + 03y \geq 0 \end{cases}$$

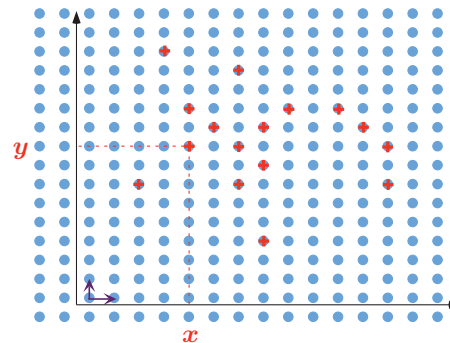
— 87 —

Effective computable approximations of an [in]finite set of points; Octagons⁵



$$\begin{cases} 1 \leq x \leq 9 \\ x + y \leq 77 \\ 1 \leq y \leq 9 \\ x - y \leq 99 \end{cases}$$

Effective computable approximations of an [in]finite set of points; Simple congruences⁷



$$\begin{cases} x = 19 \bmod 77 \\ y = 20 \bmod 99 \end{cases}$$

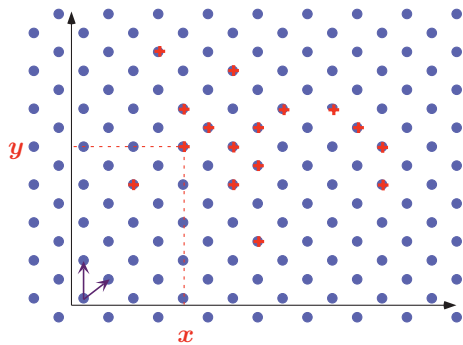
⁴ P. Cousot & R. Cousot. *Static determination of dynamic properties of programs*. Proc. 2nd Int. Symp. on Programming, Dunod, 1976.

⁵ A. Miné. *A New Numerical Abstract Domain Based on Difference-Bound Matrices*. PADO '2001. LNCS 2053, pp. 155–172. Springer 2001. See the *The Octagon Abstract Domain Library* on <http://www.di.ens.fr/~mine/oct/>

⁶ P. Cousot & N. Halbwachs. *Automatic discovery of linear restraints among variables of a program*. ACM POPL, 1978, pp. 84–97.

⁷ Ph. Granger. *Static Analysis of Arithmetical Congruences*. Int. J. Comput. Math. 30, 1989, pp. 165–190.

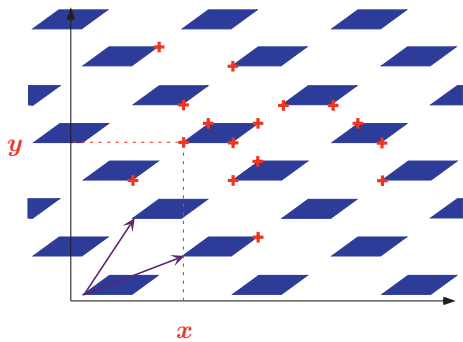
Effective computable approximations of an
[in]finite set of points; Linear
congruences⁸



$$\begin{cases} 1x + 9y = 7 \bmod 8 \\ 2x - 1y = 9 \bmod 9 \end{cases}$$

— 89 —

Effective computable approximations of an
[in]finite set of points; Trapezoidal lin-
ear congruences⁹



$$\begin{cases} 1x + 9y \in [0, 77] \bmod 10 \\ 2x - 1y \in [0, 99] \bmod 11 \end{cases}$$

⁸ Ph. Granger. *Static Analysis of Linear Congruence Equalities among Variables of a Program*. TAPSOFT '91, pp. 169–192. LNCS 493, Springer, 1991.

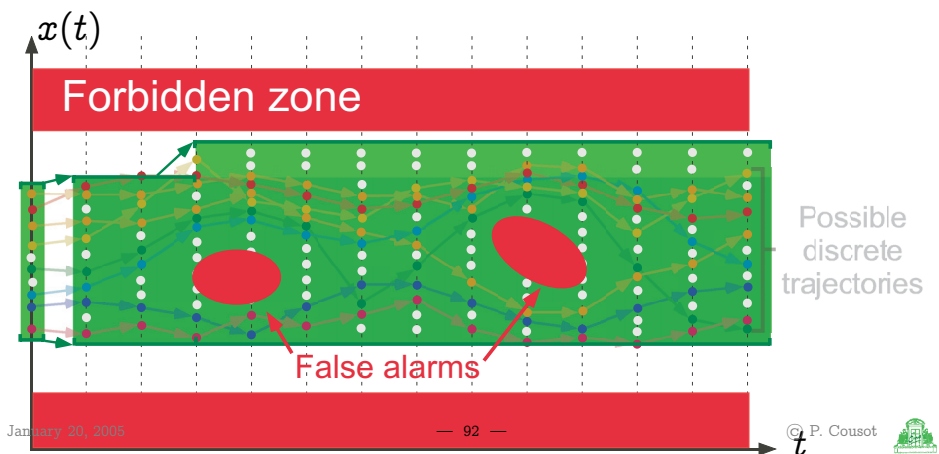
⁹ F. Masdupuy. *Array Operations Abstraction Using Semantic Analysis of Trapezoid Congruences*. ACM ICS '92.



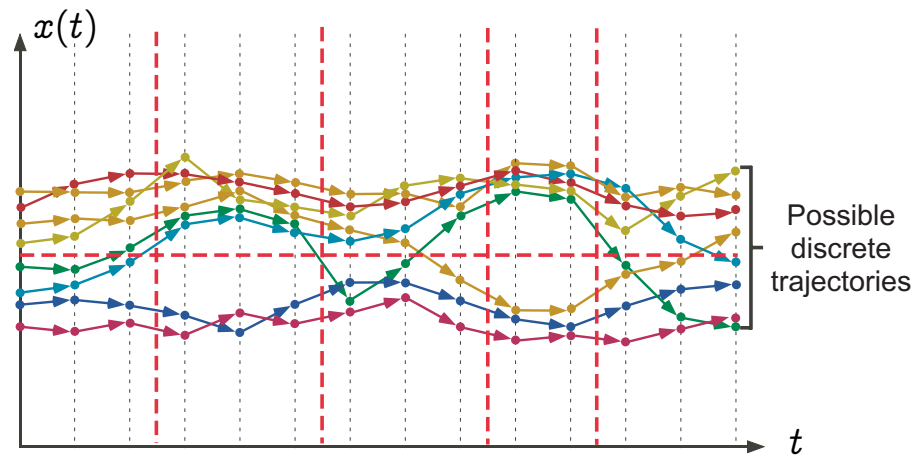
Refinement of iterates

— 91 —

Graphic example: Refinement required
by false alarms

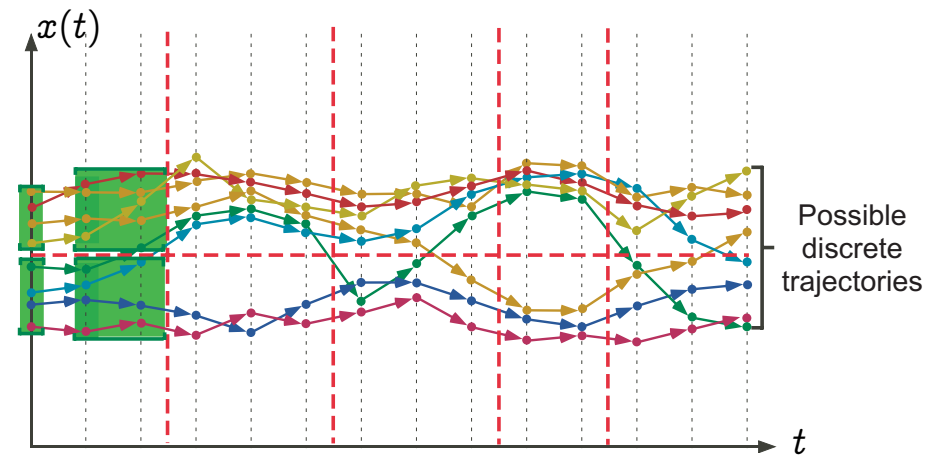


Graphic example: Partitionning

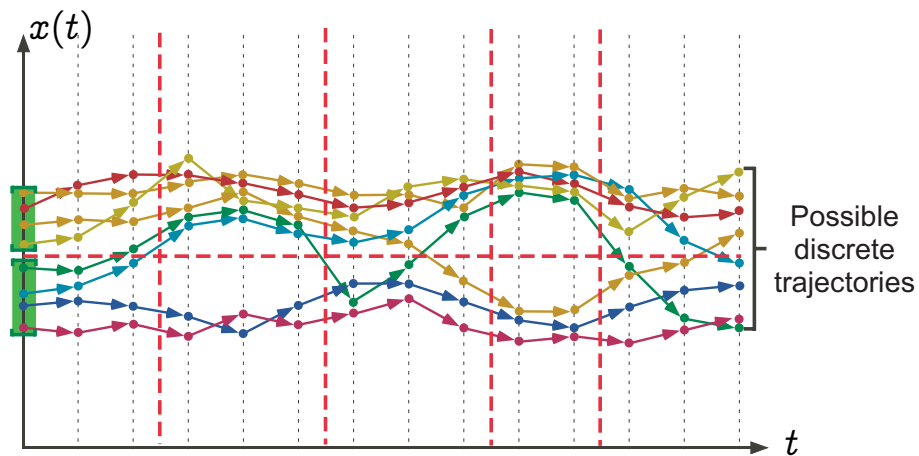


— 93 —

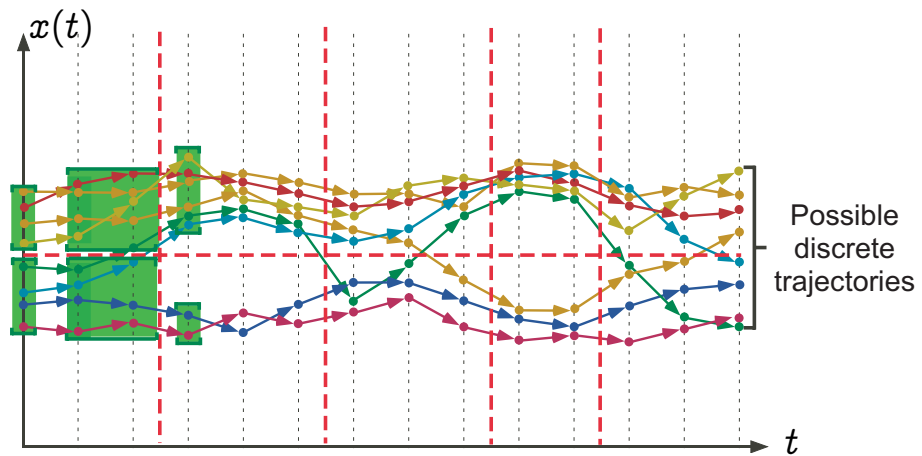
Graphic example: partitionned upward iteration with widening



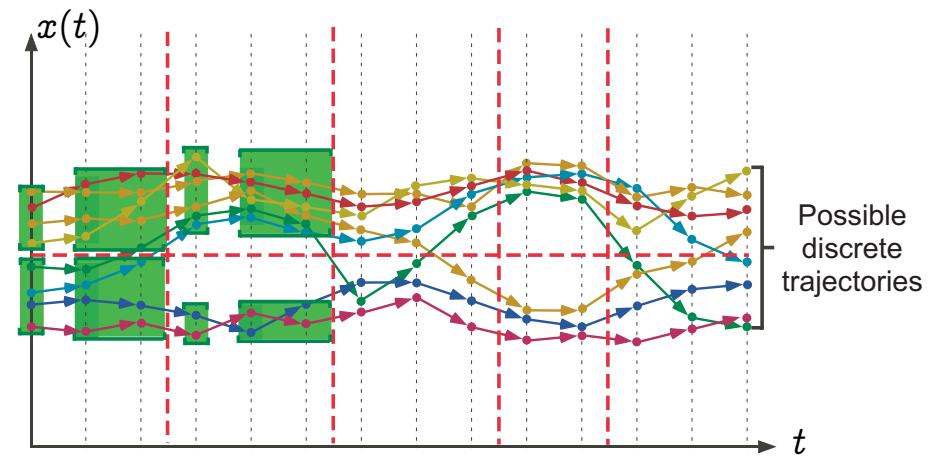
Graphic example: partitionned upward iteration with widening



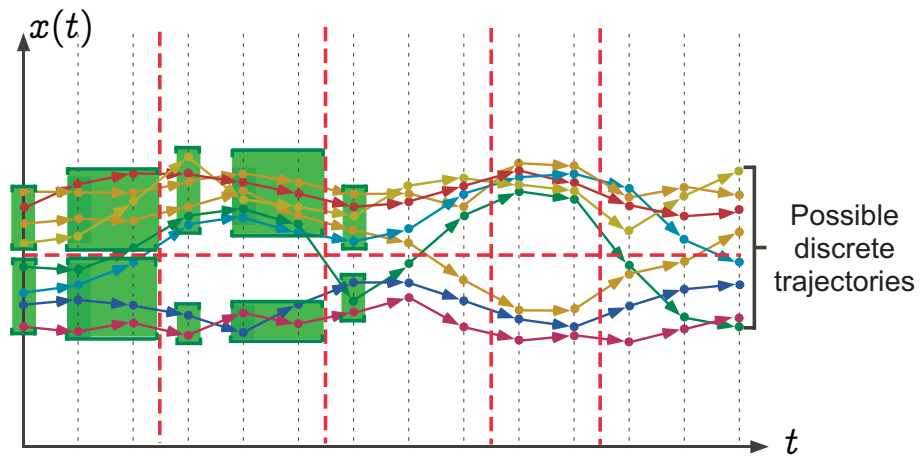
Graphic example: partitionned upward iteration with widening



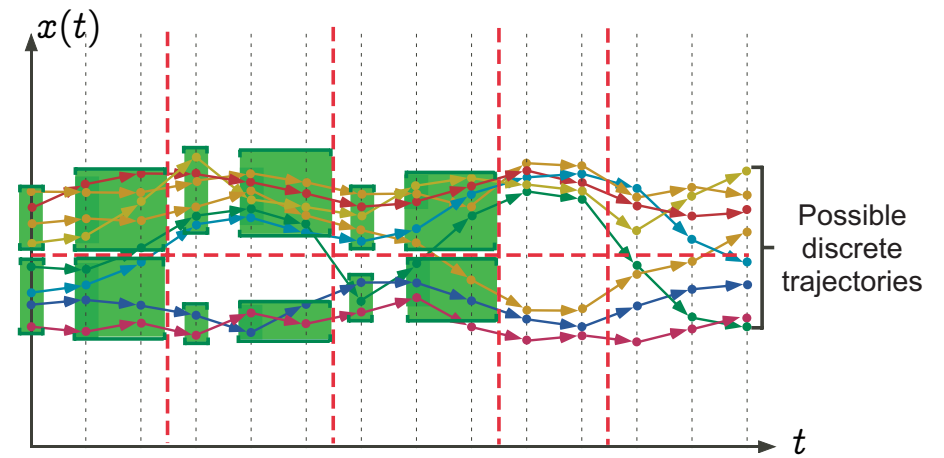
Graphic example: partitionned upward iteration with widening



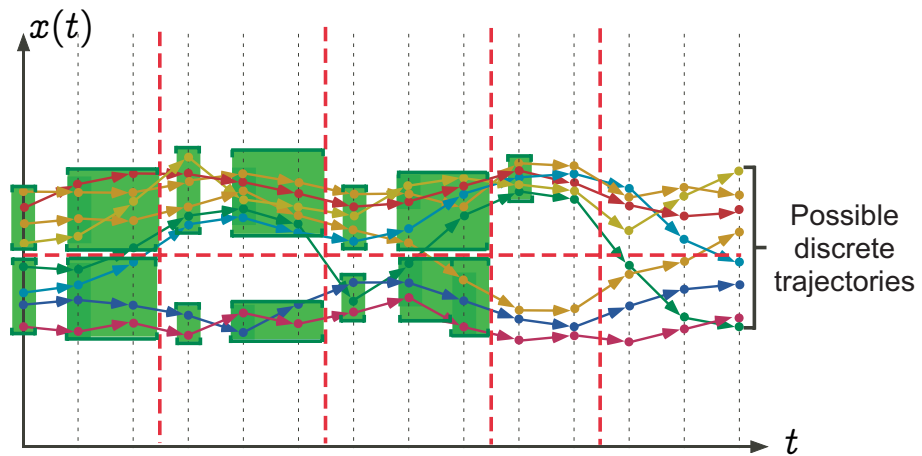
Graphic example: partitionned upward iteration with widening



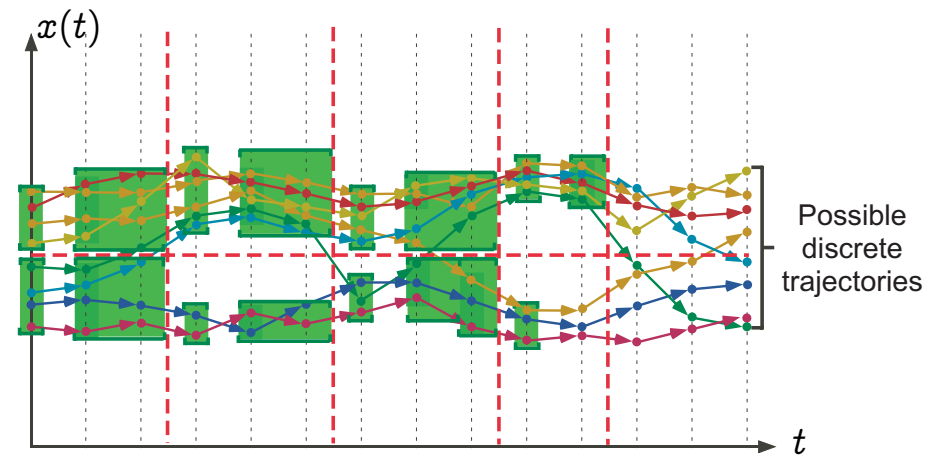
Graphic example: partitionned upward iteration with widening



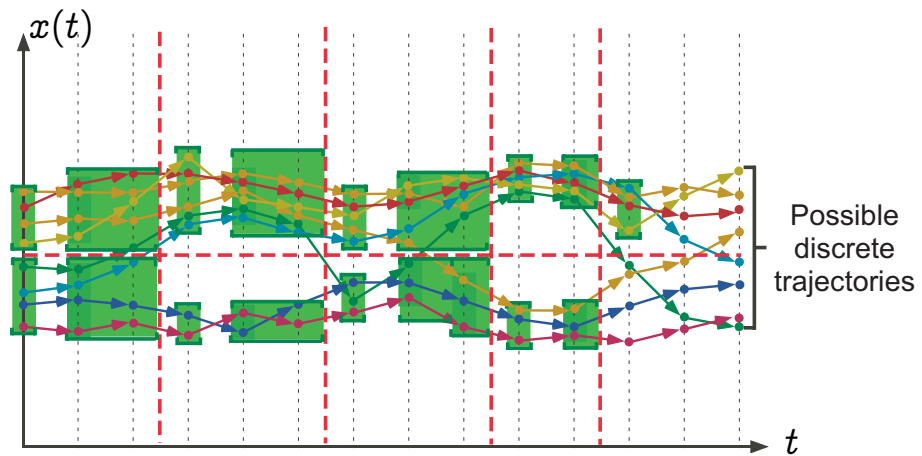
Graphic example: partitionned upward iteration with widening



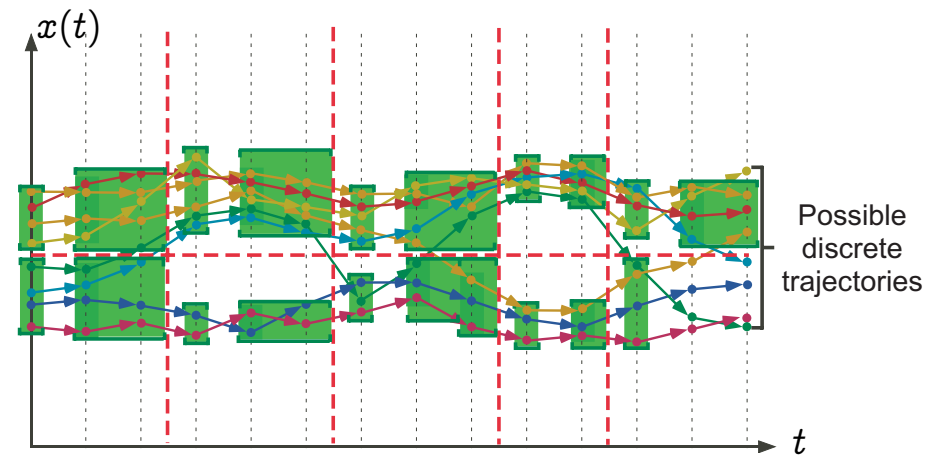
Graphic example: partitionned upward iteration with widening



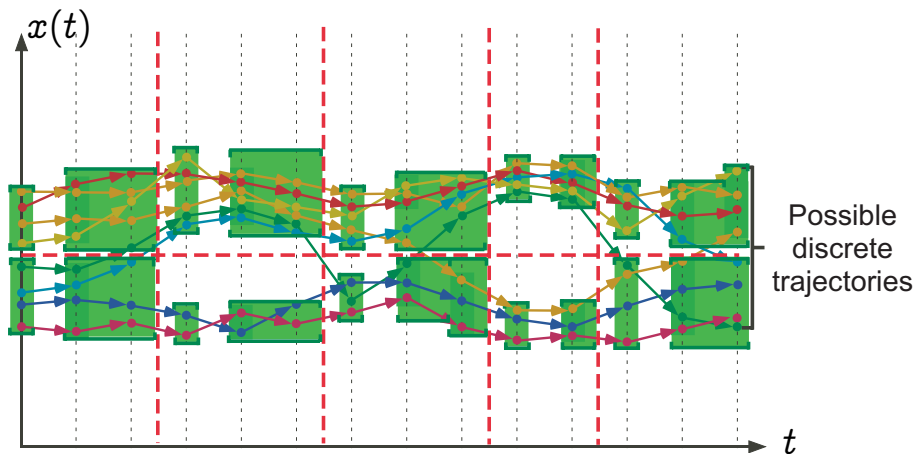
Graphic example: partitionned upward iteration with widening



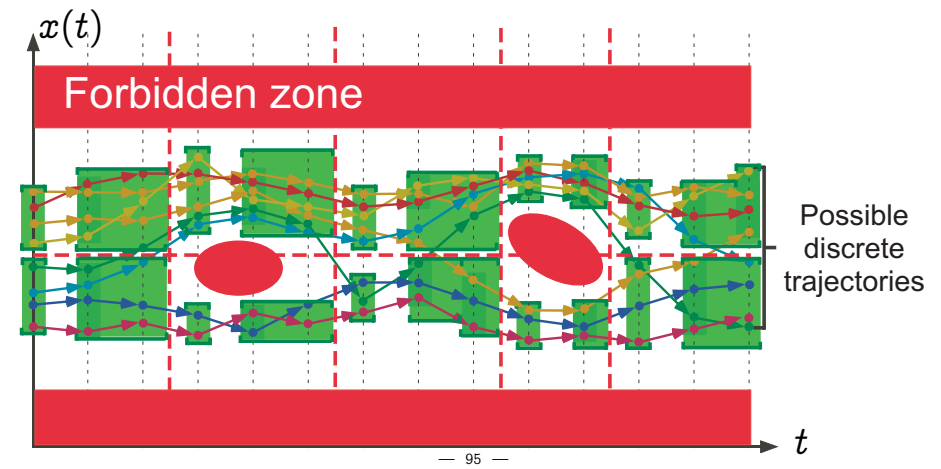
Graphic example: partitionned upward iteration with widening



Graphic example: partitionned upward iteration with widening



Graphic example: safety verification



Examples of partitionnings

- **sets of control states**: attach local information to program points instead of global information for the whole program/procedure/loop
- **sets of data states**:
 - case analysis (test, switches)
- **fixpoint iterates**:
 - widening with threshold set

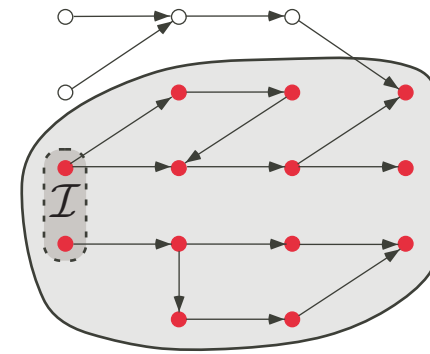
Interval widening with threshold set

- The **threshold set** T is a finite set of numbers (plus $+\infty$ and $-\infty$),
- $[a, b] \nabla_T [a', b'] = [\text{if } a' < a \text{ then } \max\{\ell \in T \mid \ell \leq a'\} \text{ else } a, \\ \text{if } b' > b \text{ then } \min\{h \in T \mid h \geq b'\} \text{ else } b]$.
- Examples (intervals):
 - sign analysis: $T = \{-\infty, 0, +\infty\}$;
 - strict sign analysis: $T = \{-\infty, -1, 0, +1, +\infty\}$;
- T is a **parameter** of the analysis.

— 97 —

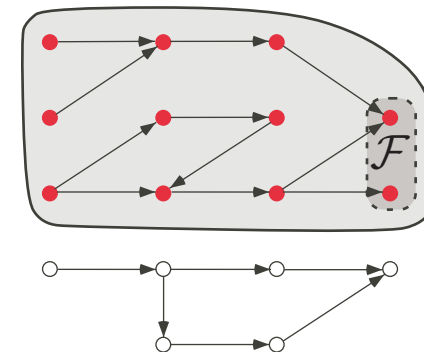
Combinations of abstractions

Forward/reachability analysis

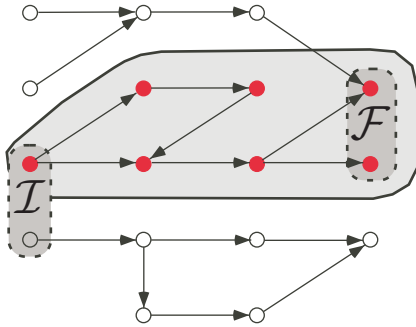


— 99 —

Backward/ancestry analysis



Iterated forward/backward analysis



— 101 —

Example of iterated forward/backward analysis

Adding an auxiliary counter k decremented in the loop body and asserted to be null on loop exit:

```
{x=y+2k, x>=y}
while (x <> y) do
  {x=y+2k, x>=y+2}
  k := k - 1;
  {x=y+2k+2, x>=y+2}
  x := x - 1;
  {x=y+2k+1, x>=y+1}
  y := y + 1
  {x=y+2k, x>=y}
od
{x=y, k=0}
assume (k = 0)
{x=y, k=0}
```

Moreover the difference of x and y must be even for proper termination

— 103 —

Example of iterated forward/backward analysis

Arithmetical mean of two integers x and y :

```
{x>=y}
while (x <> y) do
  {x>=y+2}
  x := x - 1;
  {x>=y+1}
  y := y + 1
  {x>=y}
od
{x=y}
```

Necessarily $x \geq y$ for proper termination

Bibliography




Seminal papers

- Patrick Cousot & Radhia Cousot. [Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints](#). In 4th Symp. on Principles of Programming Languages, pages 238—252. ACM Press, 1977.
- Patrick Cousot & Nicolas Halbwachs. [Automatic discovery of linear restraints among variables of a program](#). In 5th Symp. on Principles of Programming Languages, pages 84—97. ACM Press, 1978.
- Patrick Cousot & Radhia Cousot. [Systematic design of program analysis frameworks](#). In 6th Symp. on Principles of Programming Languages pages 269—282. ACM Press, 1979.

— 105 —

Recent surveys

- Patrick Cousot. [Interprétation abstraite](#). Technique et Science Informatique, Vol. 19, Nb 1-2-3. Janvier 2000, Hermès, Paris, France. pp. 155-164. 
- Patrick Cousot. [Abstract Interpretation Based Formal Methods and Future Challenges](#). In Informatics, 10 Years Back — 10 Years Ahead, R. Wilhelm (Ed.), LNCS 2000, pp. 138-156, 2001.
- Patrick Cousot & Radhia Cousot. [Abstract Interpretation Based Verification of Embedded Software: Problems and Perspectives](#). In Proc. 1st Int. Workshop on Embedded Software, EMSOFT 2001, T.A. Henzinger & C.M. Kirsch (Eds.), LNCS 2211, pp. 97–113. Springer, 2001.

Conclusion

— 107 —

Theoretical applications of abstract interpretation

- [Static Program Analysis](#) [POPL '77,78,79] including [Data-flow Analysis](#) [POPL '79,00], [Set-based Analysis](#) [FPCA '95], etc
- [Syntax Analysis](#) [TCS 290(1) 2002]
- [Hierarchies of Semantics \(including Proofs\)](#) [POPL '92, TCS 277(1-2) 2002]
- [Typing](#) [POPL '97]
- [Model Checking](#) [POPL '00]
- [Program Transformation](#) [POPL '02]
- [Software watermarking](#) [POPL '04]

Practical applications of abstract interpretation

- **Program analysis and manipulation**: a small rate of false alarms is acceptable
 - **AiT: worst case execution time** – Christian Ferdinand
- **Program verification**: no false alarms is acceptable
 - **TVLA: A system for generating abstract interpreters** – Mooly Sagiv
 - **Astrée: verification of absence of run-time errors** – Laurent Mauborgne

THE END

— 109 —

Industrial applications of abstract interpretation

- Both to **Program analysis and verification**
- **Experience with the industrial use of abstract interpretation-based static analysis tools** – Jean Souyris (Airbus France)

More references at URL www.di.ens.fr/~cousot.