On the design of widening operators

“If you widen without principles
you may converge without precision”

Nicolas Halbwachs

Verimag/CNRS
Grenoble

Automatic verification mainly consists in computing fixpoints of monotone functions on lattices.

Example: computation of reachable states

$$\text{Reach} \ = \ \text{Init} \cup \text{post}(\text{Reach}) \qquad\qquad L = 2^S$$

Model-checking = exact fixpoint computation, generally in finite or finite-depth lattices.

(Finite) abstraction [Cousot-Cousot, POPL'77]

$$L_c \underset{\gamma}{\overset{\alpha}{\rightleftharpoons}} L_a \qquad \text{(finite)}$$

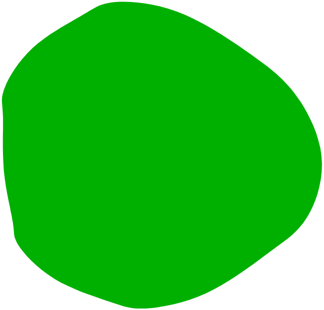
Let $F_a = \alpha \circ F_c \circ \gamma$, then $lfp(F_c) \sqsubseteq \gamma(lfp(F_a))$

$\longrightarrow$ fixpoint approximation, conservative verification

now routinely used in model-checking
[Clarke-Grumberg-Long, TOPLAS'94] [Graf-Loiseaux, CAV'93]

Example: Predicate abstraction

S, a set of states

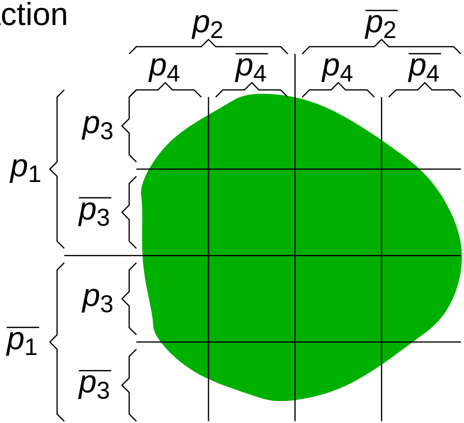


Example: Predicate abstraction

S, a set of states

P a finite set of predicates

$p_i : S \mapsto \{0, 1\}$

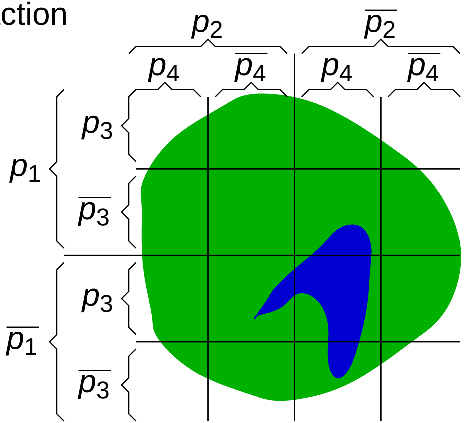


Example: Predicate abstraction

S, a set of states

P a finite set of predicates

$p_i : S \mapsto \{0,1\}$

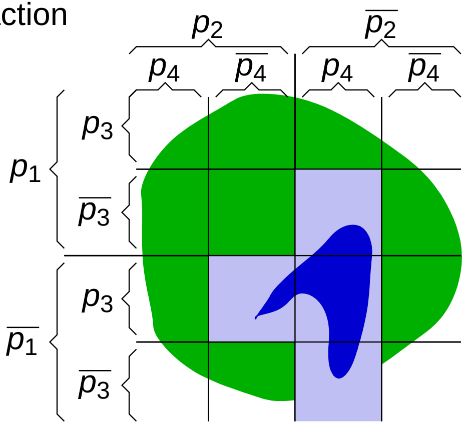


Example: Predicate abstraction

S, a set of states

P a finite set of predicates

$p_i : S \mapsto \{0,1\}$



$$L_c = 2^S \quad , \quad L_a = 2^{Mon(P)}$$

$$\alpha(X) = \{\phi \in Mon(P) \mid \exists x \in X, x \models \phi\}$$

$$\gamma(Y) = \{x \mid \exists \phi \in Y, x \models \phi\}$$

So the main remaining difference between Model-Checking and Abstract Interpretation is the use of widening.

Widening often considered as a dirty heuristic!

So the main remaining difference between Model-Checking and Abstract Interpretation is the use of widening.

Widening often considered as a dirty heuristic!

Outline of the talk

- Introduction
- Reminders about Widening
- Widening on convex polyhedra
  - Reminders about linear relation analysis
  - Classical widenings
  - Correct and incorrect attempts for improvement
  - Taking the program into account
- Avoiding widening
  - Acceleration
  - Exact abstract solution
  - Can we combine the two?

Widening: basic idea

[Cousot-Cousot, POPL'77]:

- stay in an infinite lattice
- iterative computations of $lfp(F) = \bigsqcup_{n \in \mathbb{N}} F^n(\perp)$ may be infinite
- try to guess the limit from its first terms
  $(X_0 = \perp, X_1 = F(X_0), X_2 = F(X_1) \ldots)$
- this guess is made through the computation of

$$Y_0 = X_0, \quad Y_{n+1} = Y_n \nabla F(Y_n)$$

where ∇ is a widening operator.

Widening: definition

$(L, \sqsubseteq, \sqcap, \sqcup, \bot, \top)$ a complete lattice.

$\nabla : L \times L \mapsto L$ is a widening iff

- $\forall x, y \in L,\ x \sqcup y \sqsubseteq x \nabla y$
- [chain condition]
  for all increasing chain $x_0 \sqsubseteq x_1 \sqsubseteq \ldots \sqsubseteq x_n \ldots$ in L,
  the increasing chain $y_0 = x_0, \ldots y_{n+1} = y_n \nabla x_{n+1}, \ldots$
  is not strictly increasing (i.e., stabilizes after a finite number of terms)

Widening: use

Instead of computing the (infinite) sequence

$$X_0 = \perp, X_{n+1} = F(X_n)$$

compute the finite sequence

$$Y_0 = X_0, Y_{n+1} = Y_n \nabla F(Y_n)$$

which limit is greater than $lfp(F)$

Basic example: intervals (1)

[Cousot-Cousot, ISP'76]

```
  - - - - - - - - - - - -  X0
x := 0
  - - - - - - - - - - - -  X1
while
     - - - - - - - - - -   X2
       x<100 do
     - - - - - - - - - -   X3
   x := x+1
     - - - - - - - - - -   X4
end
  - - - - - - - - - - - -  X5
```

Basic example: intervals (1)

[Cousot-Cousot, ISP'76]

- - - - - - - - - - - - $X_0 = [-\infty, +\infty]$

x := 0

- - - - - - - - - - - - $X_1 = X_0[x := 0]$

while

- - - - - - - - - - $X_2 = X_1 \sqcup X_4$

x<100 do

- - - - - - - - - - $X_3 = X_2 \sqcap [-\infty, 99]$

x := x+1

- - - - - - - - - - $X_4 = X_3[x := x + 1]$

end

- - - - - - - - - - - - $X_5 = X_2 \sqcap [100, +\infty]$

Basic example: intervals (1)

[Cousot-Cousot, ISP'76]

- - - - - - - - - - - - - $X_0 = [-\infty, +\infty]$

x := 0

- - - - - - - - - - - - - $X_1 = X_0[x := 0]$

while

- - - - - - - - - - - $X_2 = X_1 \sqcup X_4$

x<100 do

- - - - - - - - - - - $X_3 = X_2 \sqcap [-\infty, 99]$

x := x+1

- - - - - - - - - - - $X_4 = X_3[x := x + 1]$

end

- - - - - - - - - - - - - $X_5 = X_2 \sqcap [100, +\infty]$

$$X_2 = [0,0] \sqcup ((X_2 \cap [\infty, 99])\,[x := x + 1])$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99])\,[x := x+1 \Big)$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99])\, [x := x+1 \Big)$$

Exact computation

$$X_2^{(0)} \;=\; \perp$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99])\,[x := x+1] \Big)$$

Exact computation

$$\begin{array}{rcl} X_2^{(0)} & = & \bot \\ X_2^{(1)} & = & [0,0] \sqcup \bot \\ & = & [0,0] \end{array}$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99])\, [x := x+1] \Big)$$

Exact computation

$$\begin{array}{rcl}
X_2^{(0)} & = & \bot \\
X_2^{(1)} & = & [0,0] \sqcup \bot \\
 & = & [0,0] \\
X_2^{(2)} & = & [0,0] \sqcup [1,1] \\
 & = & [0,1]
\end{array}$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99])\, [x := x+1] \Big)$$

Exact computation

$$\begin{array}{rcl}
X_2^{(0)} & = & \bot \\
X_2^{(1)} & = & [0,0] \sqcup \bot \\
 & = & [0,0] \\
X_2^{(2)} & = & [0,0] \sqcup [1,1] \\
 & = & [0,1] \\
X_2^{(3)} & = & [0,0] \sqcup [1,2] \\
 & = & [0,2]
\end{array}$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99])\, [x := x+1] \Big)$$

Exact computation

$$\begin{array}{lcl}
X_2^{(0)} & = & \bot \\
X_2^{(1)} & = & [0,0] \sqcup \bot \\
 & = & {\color{blue}[0,0]} \\
X_2^{(2)} & = & [0,0] \sqcup [1,1] \\
 & = & {\color{blue}[0,1]} \\
X_2^{(3)} & = & [0,0] \sqcup [1,2] \\
 & = & {\color{blue}[0,2]} \\
 & & \ldots
\end{array}$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99]) [x := x+1] \Big)$$

Exact computation

$$\begin{aligned}
X_2^{(0)} &= \bot \\
X_2^{(1)} &= [0,0] \sqcup \bot \\
&= [0,0] \\
X_2^{(2)} &= [0,0] \sqcup [1,1] \\
&= [0,1] \\
X_2^{(3)} &= [0,0] \sqcup [1,2] \\
&= [0,2] \\
&\ldots
\end{aligned}$$

With widening

$$X_2^{(0)} = \bot$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99])\,[x := x+1] \Big)$$

Exact computation

$$\begin{array}{rcl}
X_2^{(0)} & = & \bot \\
X_2^{(1)} & = & [0,0] \sqcup \bot \\
 & = & [0,0] \\
X_2^{(2)} & = & [0,0] \sqcup [1,1] \\
 & = & [0,1] \\
X_2^{(3)} & = & [0,0] \sqcup [1,2] \\
 & = & [0,2] \\
 & & \ldots
\end{array}$$

With widening

$$\begin{array}{rcl}
X_2^{(0)} & = & \bot \\
X_2^{(1)} & = & \bot \,\nabla\, ([0,0] \sqcup \bot) \\
 & = & \bot \,\nabla\, [0,0] \\
 & = & [0,0]
\end{array}$$

Basic example: intervals (2)

$$X_2 = [0,0] \sqcup \Big((X_2 \cap [-\infty, 99]) \, [x := x+1] \Big)$$

Exact computation

$$
\begin{aligned}
X_2^{(0)} &= \perp \\
X_2^{(1)} &= [0,0] \sqcup \perp \\
&= [0,0] \\
X_2^{(2)} &= [0,0] \sqcup [1,1] \\
&= [0,1] \\
X_2^{(3)} &= [0,0] \sqcup [1,2] \\
&= [0,2] \\
& \quad \ldots
\end{aligned}
$$

With widening

$$
\begin{aligned}
X_2^{(0)} &= \perp \\
X_2^{(1)} &= \perp \nabla ([0,0] \sqcup \perp) \\
&= \perp \nabla [0,0] \\
&= [0,0] \\
X_2^{(2)} &= [0,0] \nabla ([0,0] \sqcup [1,1]) \\
&= [0,0] \nabla [0,1] \\
&= [0,\infty]
\end{aligned}
$$

convergence!

Basic example: intervals (3)

Widening on intervals:

$$\perp \nabla I = I$$

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

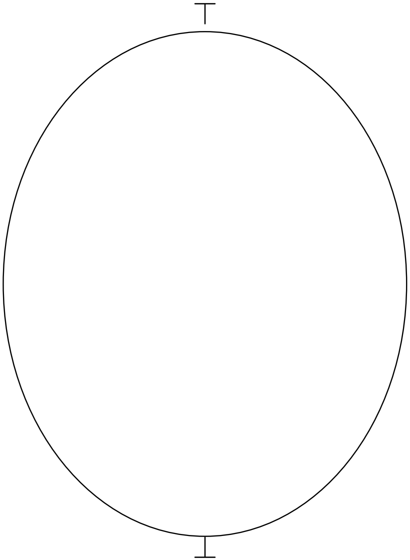
So,

$$\perp \nabla [0, 0] = [0, 0]$$

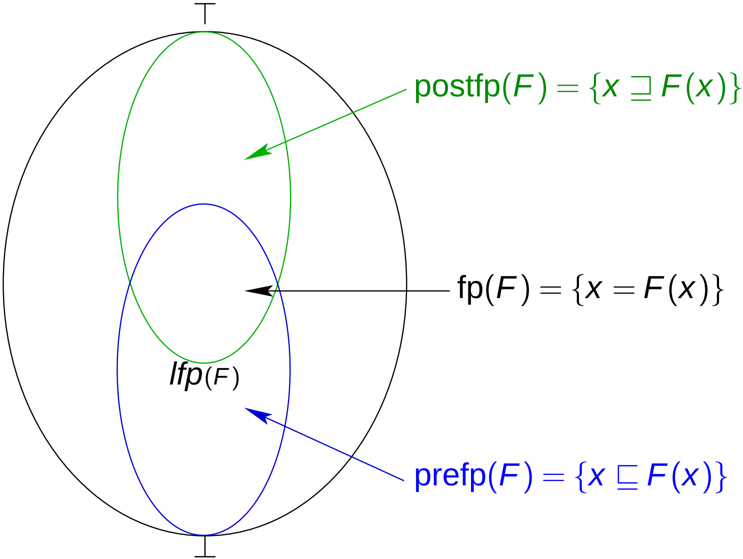
and

$$[0, 0] \nabla [0, 1] = [0, \infty]$$

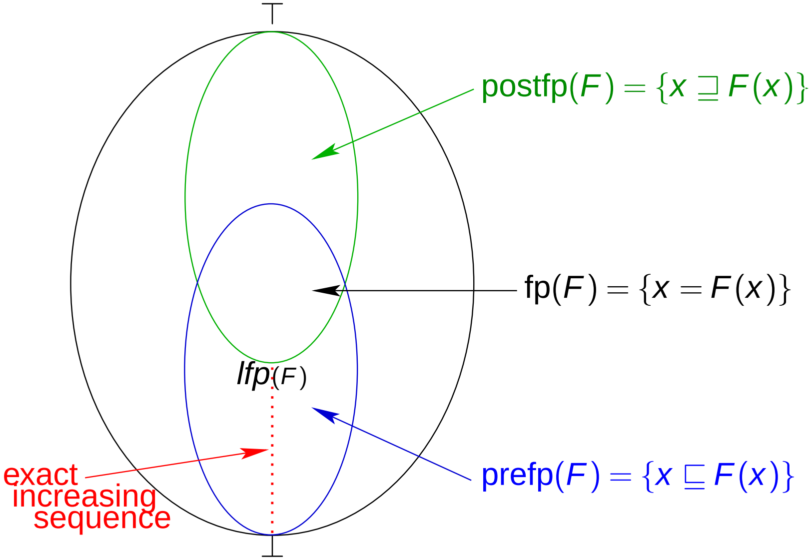
Descending sequence



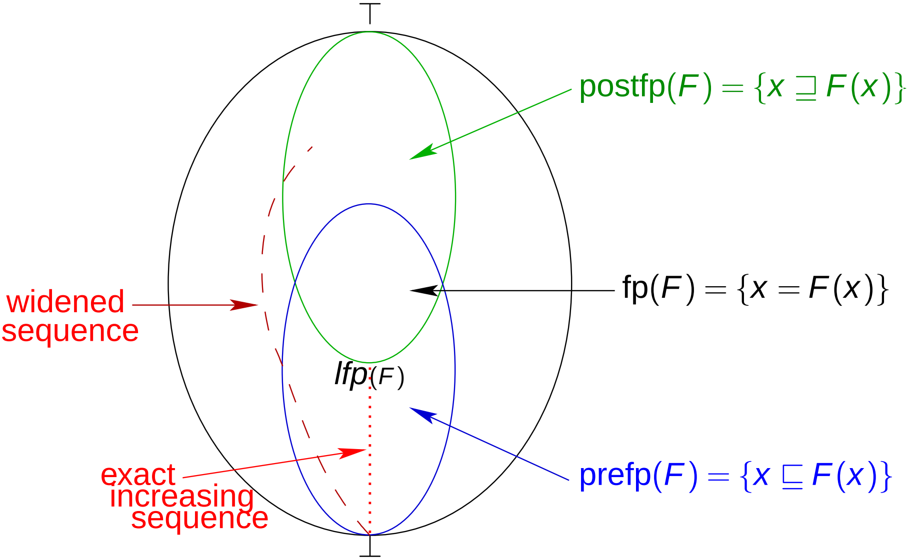
Descending sequence



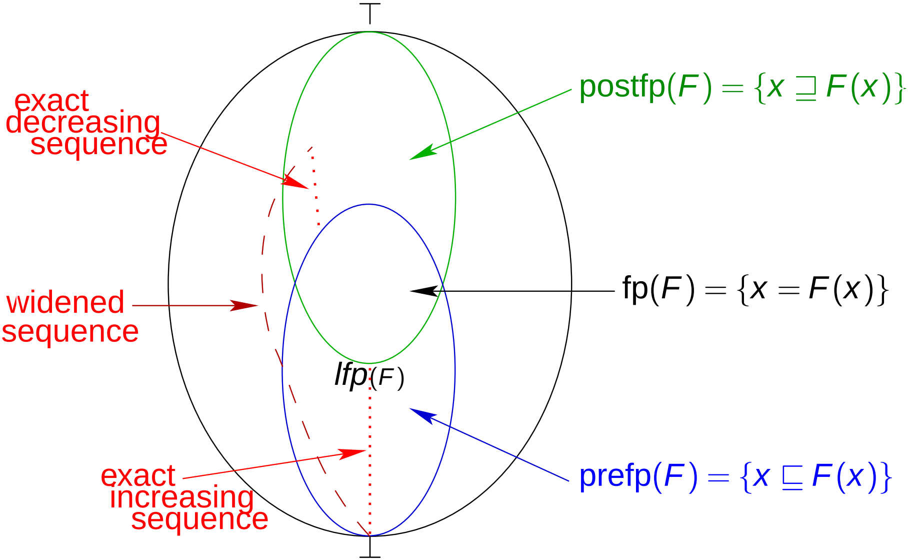
Descending sequence



Descending sequence



Descending sequence



Descending sequence: intervals

The widened sequence converged at $X_2^{(2)} = [0, \infty]$

Descending sequence:

$$
\begin{array}{rcl}
X_2^{(3)} & = & [0,0] \sqcup \left(\left(X_2^{(2)} \cap [-\infty, 99] \right) [x := x+1] \right) \\
& = & [0,0] \sqcup ([0,99][x := x+1]) \\
& = & [0,0] \sqcup [1,100] \\
& = & [0,100] \\
X_2^{(4)} & = & [0,0] \sqcup \left(\left(X_2^{(3)} \cap [-\infty, 99] \right) [x := x+1] \right) \\
& = & X_2^{(3)} \qquad \text{Fixpoint!}
\end{array}
$$

Another old example: Karp & Miller

[Karp-Miller, J. Comput. Syst. Sci. 69] Boundedness of Petri nets

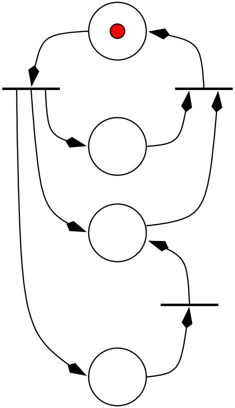
A Petri net with p places. Markings = $\mathbb{N}^p$
Order on markings: $M \preceq M' \Leftrightarrow \forall i = 1..p, M_i \leq M'_i$

> Obvious property: a set of mutually incomparable markings cannot be infinite

Enumerate the reachable markings, and whenever some marking M leads to a strictly greater marking M', replace M' by $M \nabla M'$:

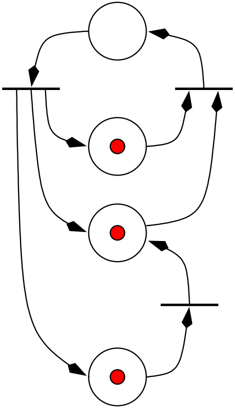
$$(M \nabla M')_i = \begin{cases} M_i & \text{if } M_i = M'_i \\ \infty & \text{if } M_i < M'_i \end{cases}$$

Karp & Miller: example



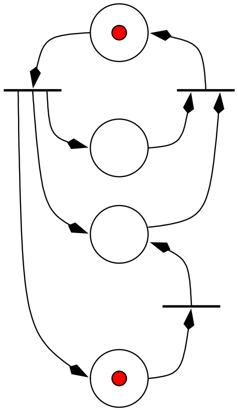
$(1, 0, 0, 0)$

Karp & Miller: example



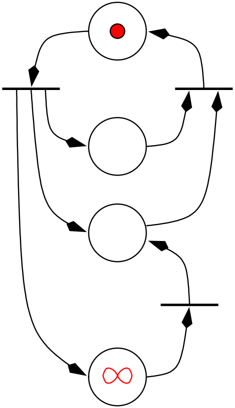
$$\begin{array}{c}(1,0,0,0)\\ \downarrow \\ (0,1,1,1)\end{array}$$

Karp & Miller: example



$$(1,0,0,0)$$
$$\downarrow$$
$$(0,1,1,1)$$
$$\downarrow$$
$$(1,0,0,1)$$

Karp & Miller: example



$$(1,0,0,0)$$

$$\downarrow$$

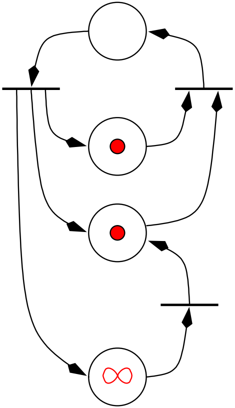
$$(0,1,1,1)$$

$$\downarrow$$

$$\cancel{(1,0,0,1)}$$

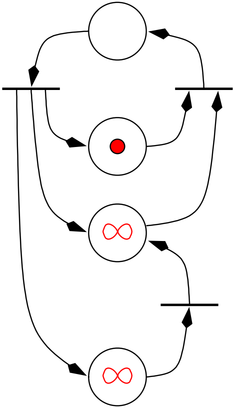
$$(1,0,0,\infty)$$

Karp & Miller: example



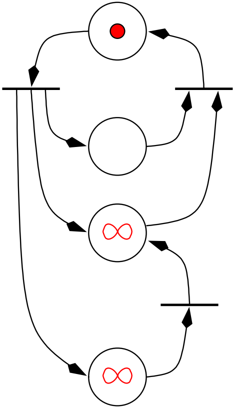
$$\begin{array}{c}(1,0,0,0)\\ \downarrow\\ (0,1,1,1)\\ \downarrow\\ \cancel{(1,0,0,1)}\\ (1,0,0,\infty)\\ \downarrow\\ (0,1,1,\infty)\end{array}$$

Karp & Miller: example



$$(1,0,0,0)$$
$$\downarrow$$
$$(0,1,1,1)$$
$$\downarrow$$
$$\cancel{(1,0,0,1)}$$
$$(1,0,0,\infty)$$
$$\downarrow$$
$$(0,1,1,\infty)$$
$$\downarrow$$
$$(0,1,\infty,\infty)$$

Karp & Miller: example



$$(1,0,0,0)$$

$$(0,1,1,1)$$

$$\cancel{(1,0,0,1)}$$
$$(1,0,0,\infty)$$

$$(0,1,1,\infty)$$

$$(0,1,\infty,\infty)$$

$$(1,0,\infty,\infty)$$

A more general definition

[Ball-Podelski-Rajamani, TACAS'02]

- Infinite states, infinite set of atomic predicates φ
- Abstract values = predicates in DNF: $\bigvee_{i \in I} \bigwedge_{j \in J_i} \varphi_{ij}$
- Widening (hint): keep in $X \nabla Y$ only the conjuncts of X which are still in Y:

$$\left(\bigvee_{i \in I} \bigwedge_{j \in J_i} \varphi_{ij}\right) \nabla \left(\bigvee_{i \in I} \bigwedge_{j \in J'_i} \varphi_{ij}\right) = \left(\bigvee_{i \in I} \bigwedge_{j \in J_i \cap J'_i} \varphi_{ij}\right)$$

Obvious need of canonical form.

Is widening useful?

- [Hankin-Hunt, ESOP'92]: All results that you get with widening can also be obtained by computing in a suitable finite lattice.

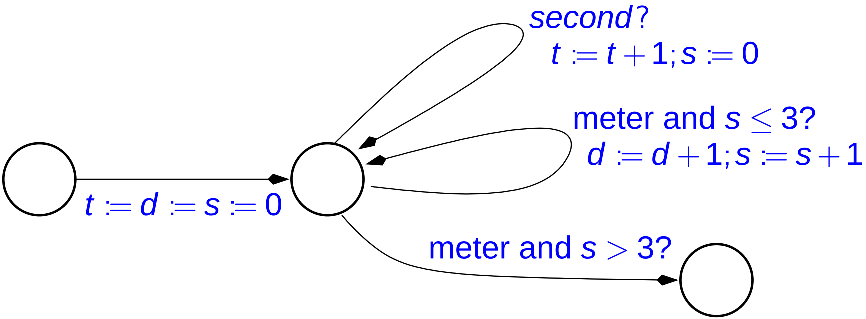
Is widening useful?

- [Hankin-Hunt, ESOP’92]: All results that you get with widening can also be obtained by computing in a suitable finite lattice.
- [Cousot-Cousot, PLILP’92]:
  - For each program, there exists a finite lattice which can be used for this program to obtain results equivalent to those obtained using widening;
  - No such finite lattice will do for all programs;
  - For a particular program, it is not possible to infer the set of needed abstract values by a simple inspection of the text of the program.

The case of convex polyhedra

Linear relation analysis: compute, in each point of a program, a set of linear inequalities invariantly satisfied by the numerical variables.

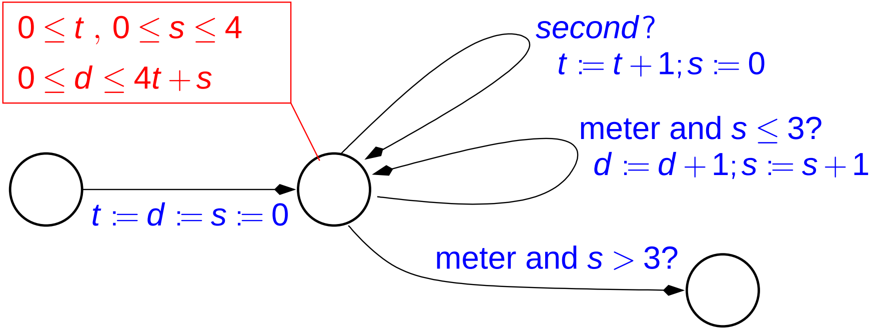
Example: a speedometer (speed limit: 4m/s)



The case of convex polyhedra

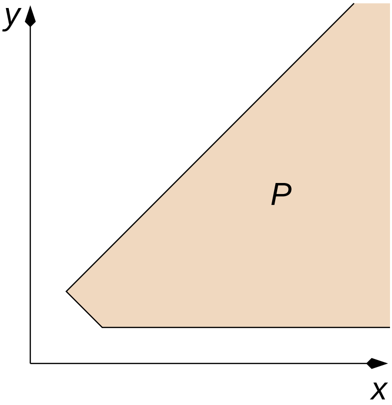
Linear relation analysis: compute, in each point of a program, a set of linear inequalities invariantly satisfied by the numerical variables.

Example: a speedometer (speed limit: 4m/s)



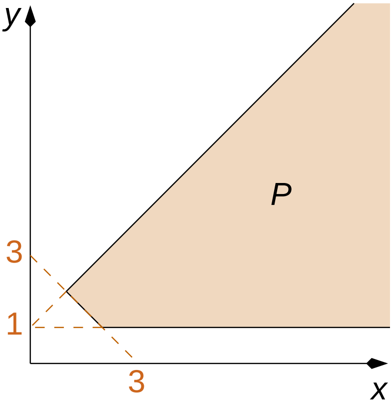
Computing over convex polyhedra (1/2)

The double representation



Computing over convex polyhedra (1/2)

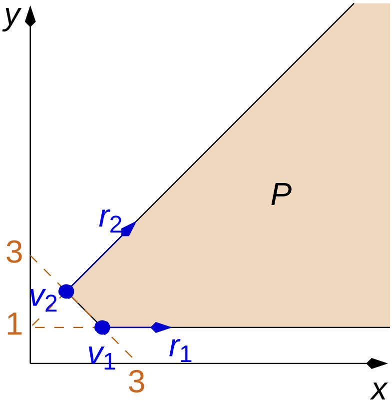
The double representation



$$\begin{aligned} P &= \{(x,y) \mid \\ &\qquad 1 \leq y \leq x+1 \wedge x+y \geq 3\} \\ &= \mathscr{C}(AX \leq b) \end{aligned}$$

Computing over convex polyhedra (1/2)

The double representation



$$P = \{(x,y) \mid 1 \leq y \leq x+1 \wedge x+y \geq 3\} = \mathscr{C}(AX \leq b)$$

$$P = \{\lambda v_1 + (1-\lambda)v_2 + \mu_1 r_1 + \mu_2 r_2 \mid \lambda \in [0,1],\ \mu_1, \mu_2 \geq 0\} = \mathscr{S}(V,R)\}$$

Computing over convex polyhedra (2/2)

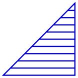
Common operations:

- intersection:
  $\mathscr{C}(AX \leq B) \cap \mathscr{C}(A'X \leq B') = \mathscr{C}(AX \leq B \wedge A'X \leq B')$
- convex hull (approximation of union):
  $\mathscr{S}(V,R) \sqcup \mathscr{S}(V',R') = \mathscr{S}(V \cup V', R \cup R')$
- affine transformation: $CP + D = \{CX + D \mid X \in P\}$
  $\mathscr{S}(C\mathscr{S}(V,R) + D) = \{Cv + D \mid v \in V\}, \{Cr \mid r \in R\}$
- test for inclusion:
  $\mathscr{S}(V,R) \subseteq \mathscr{C}(AX \leq B)$ iff $Av \leq B, \forall v \in V$ and $Ar \leq 0, \forall r \in R$
- test for emptyness: $\mathscr{S}(V,R) = \emptyset$ iff $V = \emptyset$

Standard widening (1/3)

[Cousot-Halbwachs, POPL'78]
Basic idea: keep for $P \nabla Q$ the constraints of P which are still satisfied by Q

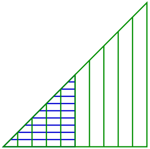
$$P = \mathscr{C}(0 \leq y \leq x \leq 1)$$



Standard widening (1/3)

[Cousot-Halbwachs, POPL'78]

Basic idea: keep for $P \nabla Q$ the constraints of P which are still satisfied by Q



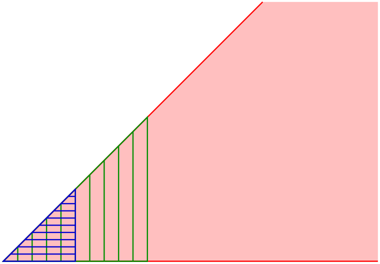
$$P = \mathscr{C}(0 \leq y \leq x \leq 1)$$

$$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$$

Standard widening (1/3)

[Cousot-Halbwachs, POPL'78]

Basic idea: keep for $P \nabla Q$ the constraints of P which are still satisfied by Q



$P = \mathscr{C}(0 \leq y \leq x \leq 1)$

$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$

$P \nabla Q = \mathscr{C}(0 \leq y \leq x)$

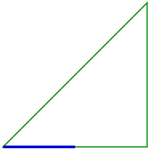
Standard widening (2/3)

Problem: None of the representations is canonical

$$P = \mathscr{C}(y = 0\ ,\ 0 \leq x \leq 1)$$

Standard widening (2/3)

Problem: None of the representations is canonical

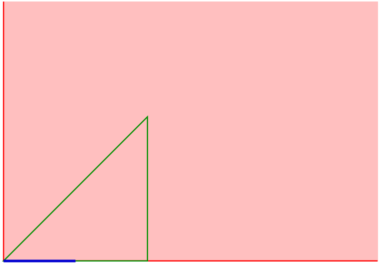


$$P = \mathscr{C}(y = 0\ ,\ 0 \leq x \leq 1)$$

$$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$$

Standard widening (2/3)

Problem: None of the representations is canonical



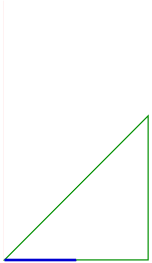
$P = \mathscr{C}(y = 0 \, , \, 0 \leq x \leq 1)$

$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$

$P \nabla Q = \mathscr{C}(0 \leq y \, , \, 0 \leq x)$

Standard widening (2/3)

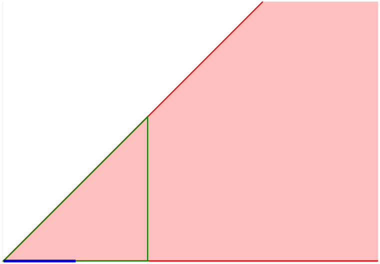
Problem: None of the representations is canonical



$$P = \cancel{\mathscr{C}(y = 0\,,\, 0 \le x \le 1)}$$
$$\mathscr{C}(0 \le y \le x \le 1\,,\, y \le 0)$$
$$Q = \mathscr{C}(0 \le y \le x \le 2)$$
$$P \nabla Q = \mathscr{C}(0 \le y\,,\, 0 \le x\,)$$

Standard widening (2/3)

Problem: None of the representations is canonical



$P = \cancel{\mathscr{C}(y = 0\,,\, 0 \leq x \leq 1)}$
$\mathscr{C}(0 \leq y \leq x \leq 1\,,\, y \leq 0)$

$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$

$P \nabla Q = \cancel{\mathscr{C}(0 \leq y\,,\, 0 \leq x)}$
$\mathscr{C}(0 \leq y \leq x)$

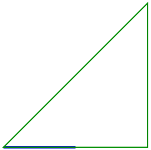
Standard widening (3/3)

Solution [Halbwachs, Thesis 1979]: keep for $P \nabla Q$ the constraints of Q which are mutually redundant with constraints of P
(i.e., can replace some constraints of P without changing it)
2 constraints are mutually redundant for P if they are saturated by the same generators of P.

$$P = \mathscr{C}(y = 0\ ,\ 0 \le x \le 1)$$

Standard widening (3/3)

Solution [Halbwachs, Thesis 1979]: keep for $P \nabla Q$ the constraints of Q which are mutually redundant with constraints of P
(i.e., can replace some constraints of P without changing it)
2 constraints are mutually redundant for P if they are saturated by the same generators of P.



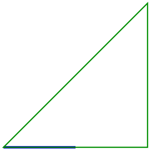
$P = \mathscr{C}(y = 0\ ,\ 0 \leq x \leq 1)$
$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$

Standard widening (3/3)

Solution [Halbwachs, Thesis 1979]: keep for $P \nabla Q$ the constraints of Q which are mutually redundant with constraints of P

(i.e., can replace some constraints of P without changing it)

2 constraints are mutually redundant for P if they are saturated by the same generators of P.



$P = \mathscr{C}(y = 0\ ,\ 0 \leq x \leq 1)$

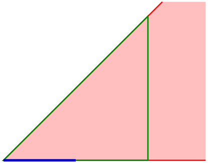
$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$

$0 \leq y$ already in P

$y \leq x$ mut. red. with $0 \leq x$ in P

Standard widening (3/3)

Solution [Halbwachs, Thesis 1979]: keep for $P \nabla Q$ the constraints of Q which are mutually redundant with constraints of P
(i.e., can replace some constraints of P without changing it)
2 constraints are mutually redundant for P if they are saturated by the same generators of P.



$P = \mathscr{C}(y = 0\ ,\ 0 \leq x \leq 1)$
$Q = \mathscr{C}(0 \leq y \leq x \leq 2)$
$0 \leq y$ already in P
$y \leq x$ mut. red. with $0 \leq x$ in P
$P \nabla Q = \mathscr{C}(0 \leq y \leq x)$

Chain condition: either the number of constraints decreases, or the dimension increases

Improving the precision

- delaying the widening
- improve the operator
- take the program into account

Delaying the widening (1/2)

[Folk!], [Halbwachs, CAV'93], [Goubault, SAS'01], [Blanchet et al., PLDI'03]

Instead of computing

$$X_0 = \bot, X_1 = X_0 \nabla F(X_0), X_2 = X_1 \nabla F(X_1) \ldots X_{n+1} = X_n \nabla F(X_n)$$

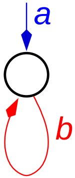
fix $k > 0$ and compute

$$X_n = \begin{cases} \bot & \text{if } n = 0 \\ F(X_{n-1}) & \text{if } n \leq k \\ X_{n-1} \nabla F(X_{n-1}) & \text{if } n > k \end{cases}$$

or, more generally, apply the widening sporadically but infinitely often.

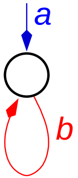
Delaying the widening (2/2)

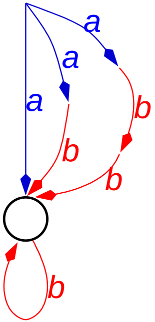
Delay, loop unrolling



Delaying the widening (2/2)

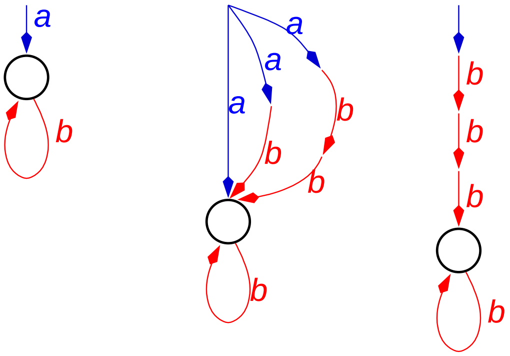
Delay, loop unrolling





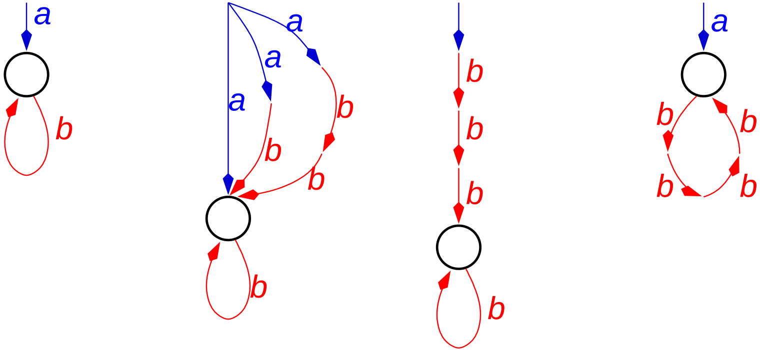
Delaying the widening (2/2)

Delay, loop unrolling



Delaying the widening (2/2)

Delay, loop unrolling



Easy, but often expensive.
The suitable number of delays or unrolling may depend on the program.

Improving the widening operator (1/4)

Correct and incorrect improvements of the standard widening on polyhedra

[Bagnara-Hill-Ricci-Zaffanella, SAS'03 and SCP'05] Hint:

$$P \nabla Q = \begin{cases} P \sqcup Q & \begin{cases} \text{if} & \dim(Q) > \dim(P) \\ \text{or} & \operatorname{codim}(Q) > \operatorname{codim}(P) \\ \text{or} & \sharp\mathscr{C}_P > \sharp\mathscr{C}_{P \sqcup Q} \\ \text{or} & \sharp V_P > \sharp V_{P \sqcup Q} \\ \text{or} & \ldots \end{cases} \\ P \nabla_S Q & \text{otherwise} \end{cases}$$

- Correct widening
- $\forall P, Q,\ P \nabla Q \subseteq P \nabla_S Q$
- *Generally* better than the standard widening.

Improving the widening operator (2/4)

- The widening is generally not monotonic.
- So, the fact that $P\nabla_1 Q \subseteq P\nabla_2 Q$ does not imply that the limit computed with ∇_1 will be better that the one provided by ∇_2.
- May slow down the convergence.
- Ideal goal: Converge fast towards a precise limit.

Improving the widening operator (3/4)

An attractive idea:

We look for the limit of the sequence $(X_n)_{n>0}$.

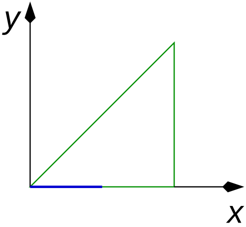
- Add a new variable k (loop counter)
- Set $X'_n = X_n \wedge (k = n)$
- express X'_n as a function of k, and let k tend to ∞.

More precisely,

- Compute $X'_1 = (X_0 \wedge k = 0) \sqcup (X_1 \wedge k = 1)$.
- Forget the constraint $k \leq 1$ in X'_1
- Eliminate k by existential quantification.

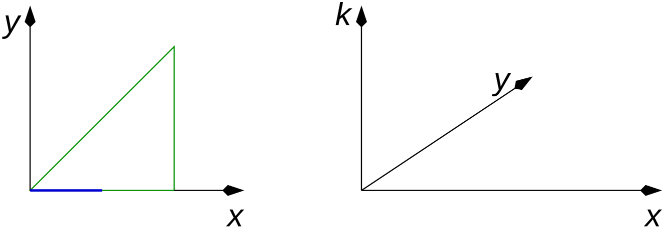
Improving the widening operator (4/4)

Example:



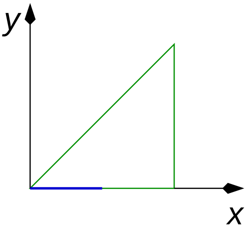
Improving the widening operator (4/4)

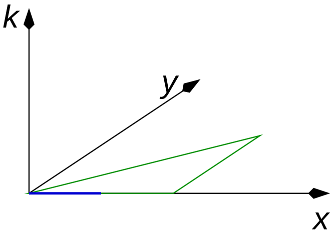
Example:



Improving the widening operator (4/4)

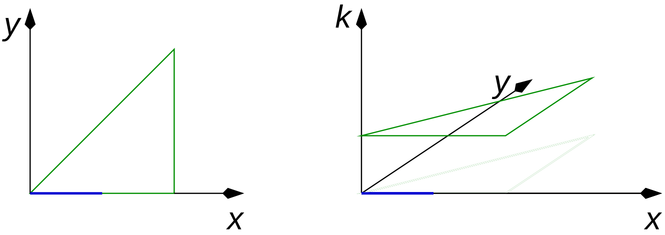
Example:





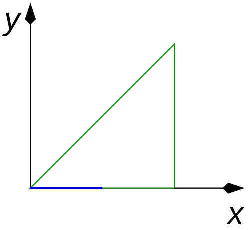
Improving the widening operator (4/4)

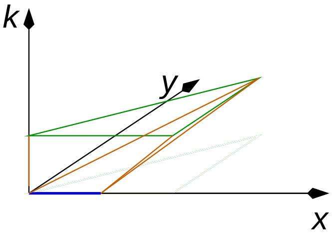
Example:



Improving the widening operator (4/4)

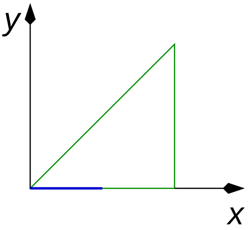
Example:

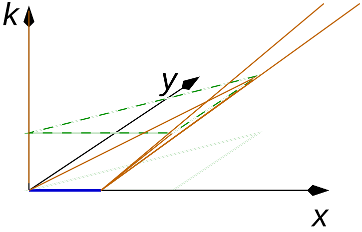




Improving the widening operator (4/4)

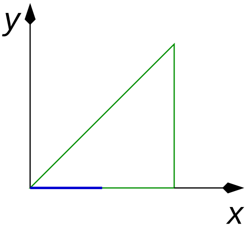
Example:

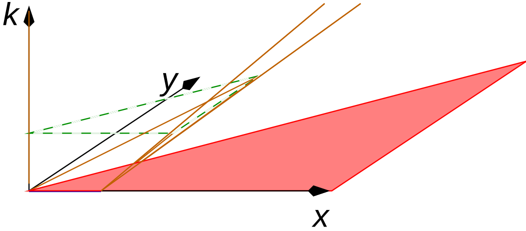




Improving the widening operator (4/4)

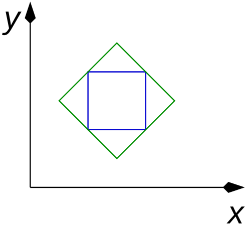
Example:





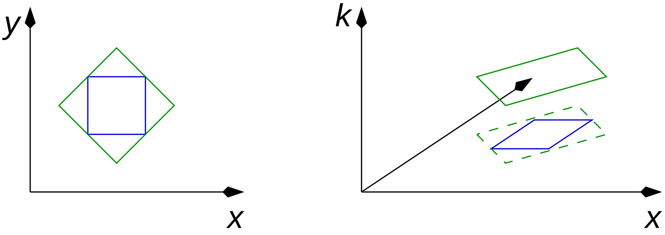
Improving the widening operator (4/5)

Unfortunately incorrect (does not fulfill the chain condition)



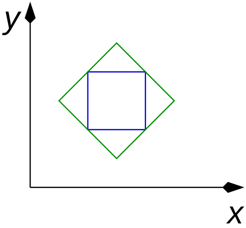
Improving the widening operator (4/5)

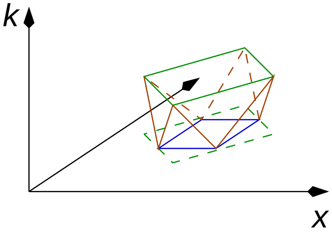
Unfortunately incorrect (does not fulfill the chain condition)



Improving the widening operator (4/5)

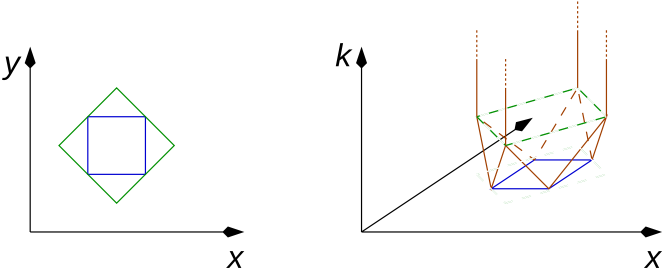
Unfortunately incorrect (does not fulfill the chain condition)





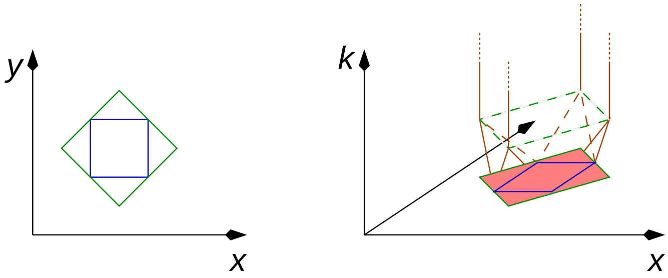
Improving the widening operator (4/5)

Unfortunately incorrect (does not fulfill the chain condition)



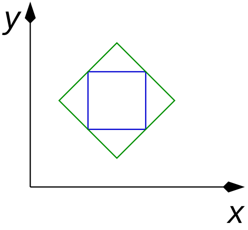
Improving the widening operator (4/5)

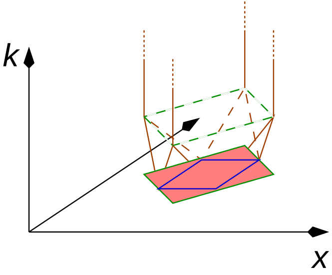
Unfortunately incorrect (does not fulfill the chain condition)



Improving the widening operator (4/5)

Unfortunately incorrect (does not fulfill the chain condition)





But adding loop counters often improves the precision of normal widening.

Taking the program into account

The widening $X \nabla F(X)$ uses the result $F(X)$ of F, but doesn't look into F.

Looking at F, i.e., at the program, can help in improving the precision:

- limited widening
- new control path
- (abstract) acceleration

Limited widening (1/2)

[Halbwachs, CAV'93], [Blanchet et al., PLDI'03]

Let C be a fixed finite set of linear constraints, then ∇^L_C defined by:

$$P \nabla^L_C Q \;=\; (P \nabla Q) \cap \mathscr{C}(\{c \in C \mid P \models c \,\wedge\, Q \models c\})$$

is a widening

Limited widening (1/2)

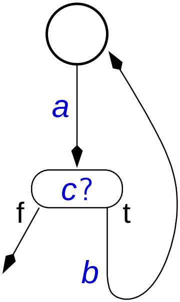
[Halbwachs, CAV'93], [Blanchet et al., PLDI'03]

Let C be a fixed finite set of linear constraints, then ∇_C^L defined by:

$$P \nabla_C^L Q = (P \nabla Q) \cap \mathscr{C}(\{c \in C \mid P \models c \wedge Q \models c\})$$

is a widening

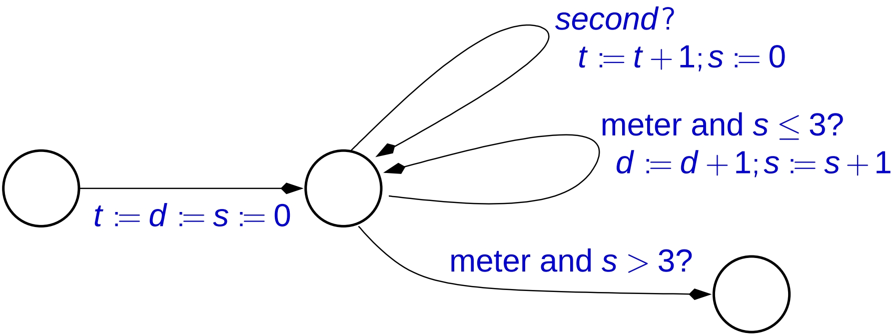
Choice of limiting constraints:



$b(c) \in C$

Limited widening (1/2)

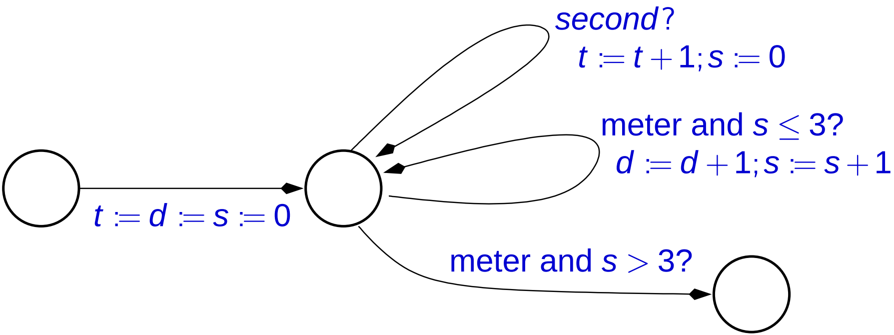
Example: speedometer



Limit the widening with $[s := s + 1](s \leq 3) = (s \leq 4)$.

Limited widening (1/2)

Example: speedometer



Limit the widening with $[s := s+1](s \leq 3) = (s \leq 4)$.

Very efficient, often makes the descending sequence useless, sometimes gives better results than the descending sequence.

New control path (1/3)

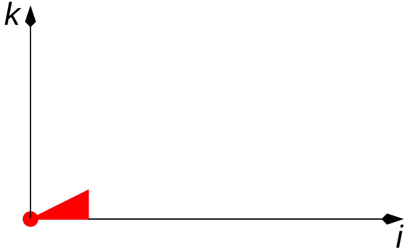
The basic assumption behind the widening is that "the program behaves regularly" (inside a loop of the program)
Not true when some path in the loop is not feasible at first iterations.

```
 i := j := 0 := k := 0;
while i'<= 100 do
   i := i+2; j := j+1;
   if ? then k := k+1;
   if j = 10 then
      j := 0; k:= k+1;
end
```

New control path (1/3)

The basic assumption behind the widening is that "the program behaves regularly" (inside a loop of the program)
Not true when some path in the loop is not feasible at first iterations.

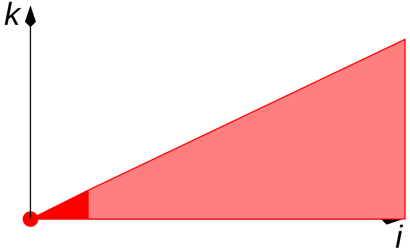
```
 i := j := 0 := k := 0;
while i'<= 100 do
   i := i+2; j := j+1;
   if ? then k := k+1;
   if j = 10 then
      j := 0; k:= k+1;
end
```



New control path (1/3)

The basic assumption behind the widening is that "the program behaves regularly" (inside a loop of the program)
Not true when some path in the loop is not feasible at first iterations.

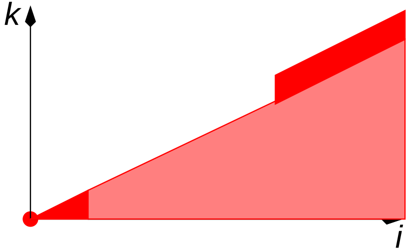
```
 i := j := 0 := k := 0;
while i'<= 100 do
   i := i+2; j := j+1;
   if ? then k := k+1;
   if j = 10 then
      j := 0; k:= k+1;
end
```



New control path (1/3)

The basic assumption behind the widening is that "the program behaves regularly" (inside a loop of the program)
Not true when some path in the loop is not feasible at first iterations.

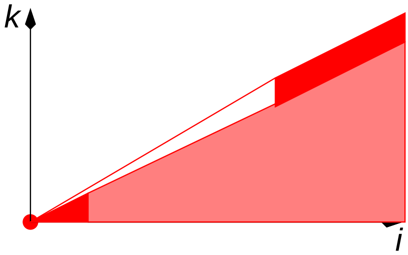
```
 i := j := 0 := k := 0;
while i'<= 100 do
   i := i+2; j := j+1;
   if ? then k := k+1;
   if j = 10 then
      j := 0; k:= k+1;
end
```



New control path (1/3)

The basic assumption behind the widening is that "the program behaves regularly" (inside a loop of the program)
Not true when some path in the loop is not feasible at first iterations.

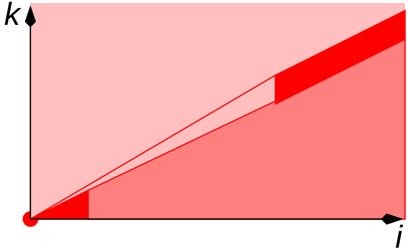
```
 i := j := 0 := k := 0;
while i'<= 100 do
   i := i+2; j := j+1;
   if ? then k := k+1;
   if j = 10 then
      j := 0; k:= k+1;
end
```



New control path (1/3)

The basic assumption behind the widening is that "the program behaves regularly" (inside a loop of the program)
Not true when some path in the loop is not feasible at first iterations.

```
 i := j := 0 := k := 0;
while i'<= 100 do
   i := i+2; j := j+1;
   if ? then k := k+1;
   if j = 10 then
      j := 0; k:= k+1;
end
```



$0 \leq i \; , \; 0 \leq k$

New control path (2/3)

Solutions: If, at step n, a previously unfeasible path in the loop becomes feasible,

- either don't widen at this step [Astrée]
- or take $X_0 \nabla X_n$, instead of $X_{n-1} \nabla X_n$ [Halbwachs, CAV'93]

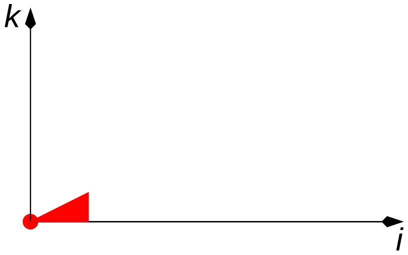
Can occur only finitely many times!

New control path (3/3)

```
 i := j := 0 := k := 0;
while i'<= 100 do
    i := i+2; j := j+1;
    if ? then k := k+1;
    if j = 10 then
        j := 0; k:= k+1;
end
```

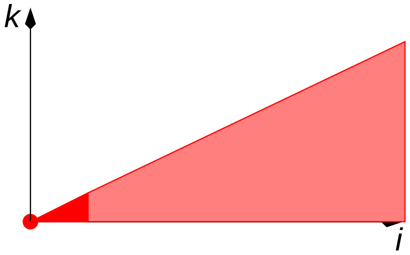
New control path (3/3)

```
 i := j := 0 := k := 0;
while i'<= 100 do
    i := i+2; j := j+1;
    if ? then k := k+1;
    if j = 10 then
        j := 0; k:= k+1;
end
```



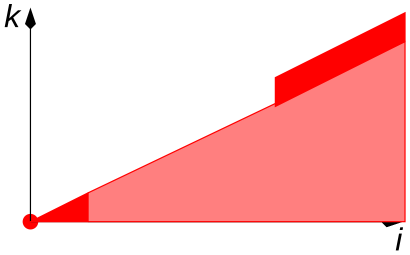
New control path (3/3)

```
 i := j := 0 := k := 0;
while i'<= 100 do
    i := i+2; j := j+1;
    if ? then k := k+1;
    if j = 10 then
        j := 0; k:= k+1;
end
```



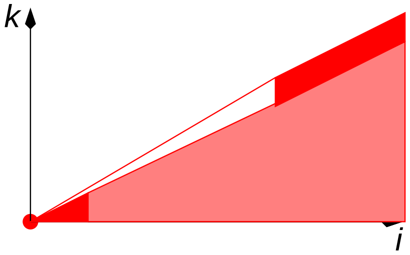
New control path (3/3)

```
 i := j := 0 := k := 0;
while i'<= 100 do
    i := i+2; j := j+1;
    if ? then k := k+1;
    if j = 10 then
        j := 0; k:= k+1;
end
```



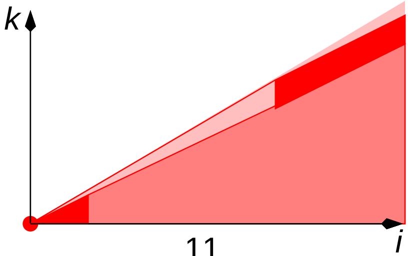
New control path (3/3)

```
 i := j := 0 := k := 0;
while i'<= 100 do
    i := i+2; j := j+1;
    if ? then k := k+1;
    if j = 10 then
        j := 0; k:= k+1;
end
```



New control path (3/3)

```
 i := j := 0 := k := 0;
while i'<= 100 do
    i := i+2; j := j+1;
    if ? then k := k+1;
    if j = 10 then
        j := 0; k:= k+1;
end
```



$$0 \leq k \leq \frac{11}{20} i$$

Exact computations

Are there cases where the effect of a loop can be computed exactly?

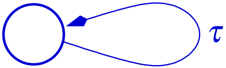
- loop acceleration
- exact abstract computation

Loop acceleration (1/3)

Exact computations in Presburger arithmetic (for a widening in Presburger arithmetic, see [Bultan-Gerber-Pugh, CAV'97])

Convergence using loop acceleration
[Boigelot-Wolper, CAV'94 and CAV'98], [Common-Jurski, CAV'98], [Finkel-Sutre, MFCS'00], [Bardin et al., CAV'03]

τ being a relation in $\mathbb{N}^n \times \mathbb{N}^n$, in which cases can we compute exactly $\tau^* = \bigcup_{k \in \mathbb{N}} \tau^k$ or $\tau^*(X_0)$?



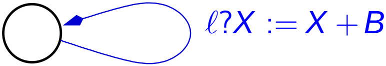
Loop acceleration (2/3)

- Restrictions on τ (e.g., $\ell(X) \wedge X' = X + B$, or $\ell(X) \wedge X' \sharp X + B$, $\sharp \in \{\leq, = \geq\}$), flat automata (without nested loops), ...
  $\longrightarrow$ exact computation in Presburger arithmetic
- Semi-algorithms for more general cases [FAST]

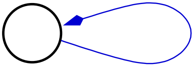
But,

- Exact computation does not scale up
- What to do with programs which don't meet the restrictions?

Loop acceleration (3/3)



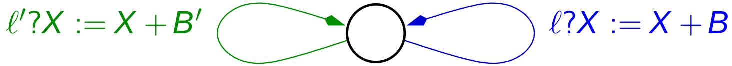
Loop acceleration (3/3)



$\tau^*(X_0) = \ell(X_0) \wedge \exists i \geq 0,\ X = X_0 + iB \wedge \ell(X_0 + (i-1)B)$

is a Presburger formula

Loop acceleration (3/3)



$\tau^*(X_0) = \ell(X_0) \wedge \exists i \geq 0,\ X = X_0 + iB \wedge \ell(X_0 + (i-1)B)$

is a Presburger formula

Not true for interleaving of transitions

Exact abstract computations (1/3)

In which cases can we solve exactly the abstract fixpoint equation?

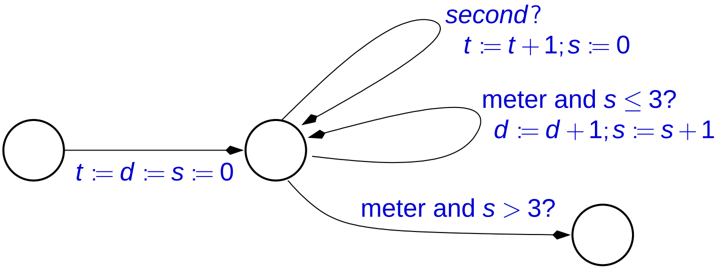
[Su-Wagner, TACAS'04]: Exact computation of the least fixpoint in interval analysis (in polynomial time)

using, basically, a suitable limited widening

Exact abstract computations (2/3)

Abstract acceleration on polyhedra [Gonnord-Halbwachs06]

Example: the speedometer in one abstract acceleration:



Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

```
t:=d:=s:=0;
while true do
   s≤3 : s++; d++
□
   true: t++; s:=0
end
```

Exact abstract computations (3/3)

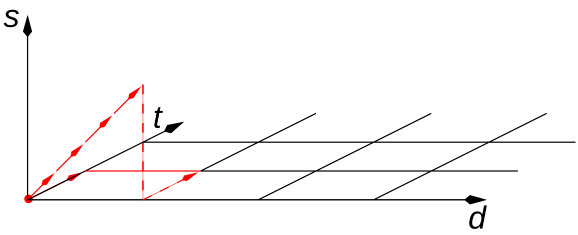
The speedometer in one abstract acceleration:

```
t:=d:=s:=0;
while true do
    s≤3 : s++; d++
□
    true: t++; s:=0
end
```

Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

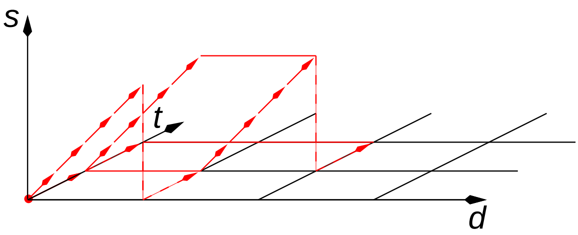
```
t:=d:=s:=0;
while true do
   s≤3 : s++; d++
□
   true: t++; s:=0
end
```



Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

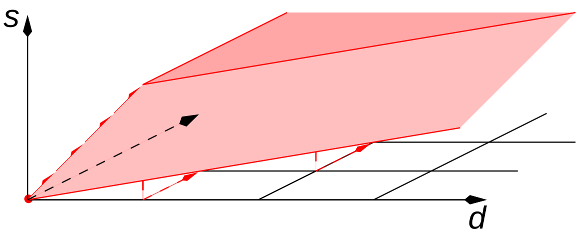
```
t:=d:=s:=0;
while true do
   s≤3 : s++; d++
□
   true: t++; s:=0
end
```



Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

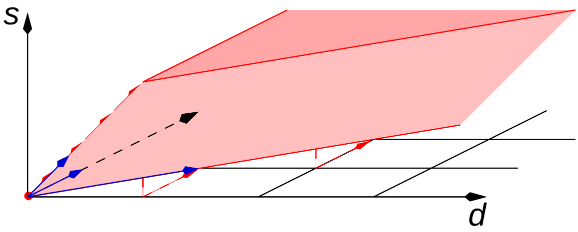
```
t:=d:=s:=0;
while true do
   s≤3 : s++; d++
□
   true: t++; s:=0
end
```



Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

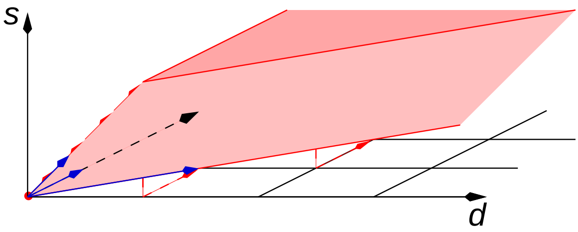
```
t:=d:=s:=0;
while true do
   s≤3 : s++; d++
□
   true: t++; s:=0
end
```



Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

```
t:=d:=s:=0;
while true do
   s≤3 : s++; d++
□
   true: t++; s:=0
end
```



$$
\begin{aligned}
P &= (\{(0,0,0)\} \nearrow \{(1,0,0),(0,1,1),(1,4,0)\}) \cap (s \leq 4) \\
&= (t \geq 0, 0 \leq s \leq 4, 0 \leq d \leq 4t + s)
\end{aligned}
$$

Exact abstract computations (4/3)

Obvious advantages:

- when it applies, it is both more precise and more efficient
- easy to combine with widening

Conclusions

- Widening is more than a dirty heuristic
  - it is based on a reasonable hypothesis:
    Regular, foreseeable programs should be easy to analyse
  - It allows to work in quite expressive lattices.
  - It allows general programs to be analysed.
- It may be more effective to use well-chosen application strategies, (e.g., limited widening, specific cases when new pathes are discovered), than to endlessly look for “better” widening.
- Looking at the program can lead to significant improvements.