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

$$\text{Reach} = \text{Init} \cup \text{post(Reach)}$$

$$L = 2^S$$

Model-checking = exact fixpoint computation, generally in finite or finite-depth lattices.
(Finite) abstraction [Cousot-Cousot, POPL’77]

\[ L_c \xrightarrow{\alpha} L_a \quad \text{(finite)} \]

Let \( F_a = \alpha \circ F_c \circ \gamma \), then \( \text{lfp}(F_c) \sqsubseteq \gamma(\text{lfp}(F_a)) \)

\( \rightarrow \) 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

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

$L_c = 2^S$, \hspace{1cm} $L_a = 2^{\text{Mon}(P)}$

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

$\gamma(Y) = \{ x \mid \exists \phi \in Y, x \models \phi \}$
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 \( \text{lfp}(F) = \bigsqcup_{n \in \mathbb{N}} F^n(\perp) \) 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

\[
Y_0 = X_0, \quad Y_{n+1} = Y_n \triangledown F(Y_n)
\]

where \( \triangledown \) is a widening operator.
Widening: definition

\((L, \subseteq, \cap, \cup, \bot, \top)\) a complete lattice.

\(\nabla : L \times L \mapsto L\) is a widening iff

- \(\forall x, y \in L, x \sqcup y \subseteq x \nabla y\)

- [chain condition]
  for all increasing chain \(x_0 \subseteq x_1 \subseteq \ldots \subseteq x_n \ldots\) 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)
Reminders about widening

Widening: use

Instead of computing the (infinite) sequence

\[ X_0 = \bot, \, X_{n+1} = F(X_n) \]

compute the finite sequence

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

which limit is greater than \( \text{lfp}(F) \)
Basic example: intervals (1)

[Cousot-Cousot, ISP’76]

\[ x := 0 \]

\[ \text{while} \]

\[ x < 100 \text{ do} \]

\[ x := x + 1 \]

\[ \text{end} \]

\[ X_0 \]

\[ X_1 \]

\[ X_2 \]

\[ X_3 \]

\[ X_4 \]

\[ X_5 \]
Basic example: intervals (1)

[Cousot-Cousot, ISP’76]

\[ x = 0 \]
\[ x < 100 \text{ do} \]
\[ x := x + 1 \]

\[ X_0 = [-\infty, +\infty] \]
\[ X_1 = X_0[x := 0] \]
\[ X_2 = X_1 \sqcup X_4 \]
\[ X_3 = X_2 \cap [-\infty, 99] \]
\[ X_4 = X_3[x := x + 1] \]
\[ X_5 = X_2 \cap [100, +\infty] \]
Basic example: intervals (1)

[Cousot-Cousot, ISP’76]

\[
X_0 = [-\infty, +\infty]
\]
\[
x := 0
\]
\[
X_1 = X_0[x := 0]
\]
while
\[
x < 100 \text{ do}
\]
\[
X_2 = X_1 \sqcup X_4
\]
\[
x := x + 1
\]
\[
X_3 = X_2 \cap [-\infty, 99]
\]
\[
X_4 = X_3[x := x + 1]
\]
end
\[
X_5 = X_2 \cap [100, +\infty]
\]
\[
X_2 = [0, 0] \sqcup ((X_2 \cap [\infty, 99]) \sqcup [x := x + 1])
\]
Basic example: intervals (2)

\[ X_2 = [0, 0] \sqcup \left( (X_2 \cap [-\infty, 99]) [x := x + 1] \right) \]
Basic example: intervals (2)

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

Exact computation

\[ X_2^{(0)} = \bot \]
Basic example: intervals (2)

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

Exact computation

\[
\begin{align*}
X_2^{(0)} &= ⊥ \\
X_2^{(1)} &= [0, 0] \sqcup ⊥ \\
&= [0, 0]
\end{align*}
\]
Basic example: intervals (2)

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

Exact computation

\[
\begin{align*}
X_2^{(0)} &= \bot \\
X_2^{(1)} &= [0, 0] \sqcup \bot \\
&= [0, 0] \\
X_2^{(2)} &= [0, 0] \sqcup [1, 1] \\
&= [0, 1]
\end{align*}
\]
Basic example: intervals (2)

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

Exact computation

\[
\begin{align*}
X_2^{(0)} &= \bot \\
X_2^{(1)} &= [0, 0] \sqcup \bot \\
 &= [0, 0] \\
X_2^{(2)} &= [0, 0] \sqcup [1, 1] \\
 &= [0, 1] \\
X_2^{(3)} &= [0, 0] \sqcup [1, 2] \\
 &= [0, 2]
\end{align*}
\]
Basic example: intervals (2)

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

Exact computation

\[
\begin{align*}
X_2^{(0)} & = \bot \\
X_2^{(1)} & = [0, 0] \sqcup \bot \\
& = [0, 0] \\
X_2^{(2)} & = [0, 0] \sqcup [1, 1] \\
& = [0, 1] \\
X_2^{(3)} & = [0, 0] \sqcup [1, 2] \\
& = [0, 2] \\
\ldots
\end{align*}
\]
Basic example: intervals (2)

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

Exact computation

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

With widening

\[
\begin{align*}
X_2^{(0)} &= \bot \\
X_2 &= \bot
\end{align*}
\]
Basic example: intervals (2)

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

Exact computation

\begin{align*}
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{align*}

With widening

\begin{align*}
X_2^{(0)} & = \bot \\
X_2^{(1)} & = \bot \nabla ([0, 0] \sqcup \bot) \\
& = \bot \nabla [0, 0] \\
& = [0, 0]
\end{align*}
Basic example: intervals (2)

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

**Exact computation**

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

**With widening**

\[
\begin{align*}
X_2^{(0)} &= \bot \\
X_2^{(1)} &= \bot \triangledown ([0, 0] \sqcup \bot) \\
&= \bot \triangledown [0, 0] \\
&= [0, 0] \\
X_2^{(2)} &= [0, 0] \triangledown ([0, 0] \sqcup [1, 1]) \\
&= [0, 0] \triangledown [0, 1] \\
&= [0, \infty] \\
\end{align*}
\]

convergence!
Basic example: intervals (3)

Widening on intervals:

\[
\bot \triangledown I = I
\]

\[
[a, b] \triangledown [c, d] = \begin{cases} 
\text{if } c < a \text{ then } -\infty \text{ else } a, & \text{if } d > b \text{ then } \infty \text{ else } b 
\end{cases}
\]

So,

\[
\bot \triangledown [0, 0] = [0, 0]
\]

and

\[
[0, 0] \triangledown [0, 1] = [0, \infty]
\]
Descending sequence
Descending sequence

\[ \text{postfp}(F) = \{ x \sqsupseteq F(x) \} \]

\[ \text{fp}(F) = \{ x = F(x) \} \]

\[ \text{lfp}(F) \]

\[ \text{prefp}(F) = \{ x \sqsubseteq F(x) \} \]
Descending sequence

\[ \text{postfp}(F) = \{ x \sqsupseteq F(x) \} \]

\[ \text{fp}(F) = \{ x = F(x) \} \]

\[ \text{prefp}(F) = \{ x \sqsubseteq F(x) \} \]

exact increasing sequence

\(\text{Ifp}(F)\)
**Descending sequence**

\[ \text{postfp}(F) = \{ x \sqsupseteq F(x) \} \]

\[ \text{fp}(F) = \{ x = F(x) \} \]

\[ \text{prefp}(F) = \{ x \sqsubseteq F(x) \} \]

**Reminders about widening**

N. Halbwachs (Verimag/CNRS)  On the design of widening operators  12 / 46
Reminders about widening

Descending sequence

\[ \text{postfp}(F) = \{ x \sqsupseteq F(x) \} \]
\[ \text{fp}(F) = \{ x = F(x) \} \]
\[ \text{lfp}(F) \]
\[ \text{prefp}(F) = \{ x \sqsubseteq F(x) \} \]

exact increasing sequence

widened sequence

exact decreasing sequence
Reminders about widening

**Descending sequence: intervals**

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

Descending sequence:

\[
X_2^{(3)} = [0, 0] \sqcup \left( \left( X_2^{(2)} \cap [-\infty, 99] \right) [x := x + 1] \right) \\
= [0, 0] \sqcup ([0, 99][x := x + 1]) \\
= [0, 0] \sqcup [1, 100] \\
= [0, 100]
\]

\[
X_2^{(4)} = [0, 0] \sqcup \left( \left( X_2^{(3)} \cap [-\infty, 99] \right) [x := x + 1] \right) \\
= X_2^{(3)} \quad \text{Fixpoint!}
\]
Another old example: Karp & Miller


A Petri net with \( p \) places. Markings = \( \mathbb{N}^p \)

Order on markings: \( M \preceq M' \iff \forall i = 1..p, M_i \leq M'_i \)

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 \( M \vee M' \):

\[
(M \vee M')_i = \begin{cases} 
M_i & \text{if } M_i = M'_i \\
\infty & \text{if } M_i < M'_i
\end{cases}
\]
Karp & Miller: example

\[(1, 0, 0, 0)\]
Karp & Miller: example

(1,0,0,0)
(0,1,1,1)
Karp & Miller: example

(1, 0, 0, 0)

(0, 1, 1, 1)

(1, 0, 0, 1)
Karp & Miller: example

\[(1, 0, 0, 0)\]
\[(0, 1, 1, 1)\]
\[(1, 0, 0, 1)\]
\[(1, 0, 0, \infty)\]
Karp & Miller: example

\[(1, 0, 0, 0)\]
\[(0, 1, 1, 1)\]
\[(1, 0, 0, 1)\]
\[(1, 0, 0, \infty)\]
\[(0, 1, 1, \infty)\]
Karp & Miller: example

\[(1, 0, 0, 0)\]
\[(0, 1, 1, 1)\]
\[(1, 0, 0, 1)\]
\[(1, 0, 0, \infty)\]
\[(0, 1, 1, \infty)\]
\[(0, 1, \infty, \infty)\]
Karp & Miller: example
Reminders about widening

A more general definition

[Ball-Podelski-Rajamani, TACAS’02]

- Infinite states, infinite set of atomic predicates $\varphi$
- Abstract values $= \text{predicates in DNF}: \bigvee_{i \in I} \bigwedge_{j \in J_i} \varphi_{ij}$
- 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)

\[ 0 \leq t, \ 0 \leq s \leq 4 \]
\[ 0 \leq d \leq 4t + s \]

\[ t := t + 1; s := 0 \]
\[ d := d + 1; s := s + 1 \]
Computing over convex polyhedra (1/2)

The double representation

\[ P = \{ (x, y) \mid 1 \leq y \leq x + 1 \land x + y \geq 3 \} = C(AX \leq b) \]

\[ P = \{ \lambda v_1 + (1 - \lambda)v_2 + \mu_1 r_1 + \mu_2 r_2 \mid \lambda \in [0, 1], \mu_1, \mu_2 \geq 0 \} = S(V, R) \]
Computing over convex polyhedra (1/2)

The double representation

\[ P = \{(x, y) \mid 1 \leq y \leq x + 1 \land x + y \geq 3\} = \mathcal{C}(AX \leq b) \]
Computing over convex polyhedra (1/2)

The double representation

\[ P = \{(x, y) \mid 1 \leq y \leq x + 1 \land x + y \geq 3\} = C(AX \leq b) \]

\[ P = \{\lambda v_1 + (1 - \lambda)v_2 + \mu_1 r_1 + \mu_2 r_2 \mid \lambda \in [0, 1], \mu_1, \mu_2 \geq 0\} = \mathcal{I}(V, R) \]
Computing over convex polyhedra (2/2)

Common operations:

- intersection:
  \[ \mathcal{C}(AX \leq B) \cap \mathcal{C}(A'X \leq B') = \mathcal{C}(AX \leq B \land A'X \leq B') \]

- convex hull (approximation of union):
  \[ \mathcal{S}(V, R) \cup \mathcal{S}(V', R') = \mathcal{S}(V \cup V', R \cup R') \]

- affine transformation:
  \[ CP + D = \{ CX + D \mid X \in P \} \]
  \[ \mathcal{S}(C \mathcal{S}(V, R) + D) = \{ Cv + D \mid v \in V \}, \{ Cr \mid r \in R \} \]

- test for inclusion:
  \[ \mathcal{S}(V, R) \subseteq \mathcal{C}(AX \leq B) \text{ iff } Av \leq B, \forall v \in V \text{ and } Ar \leq 0, \forall r \in R \]

- test for emptiness:
  \[ \mathcal{S}(V, R) = \emptyset \text{ iff } V = \emptyset \]
Standard widening (1/3)

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

$$P = \mathcal{C}(0 \leq y \leq x \leq 1)$$
Standard widening (1/3)

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

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

$$Q = C(0 \leq y \leq x \leq 2)$$
Standard widening (1/3)

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

\[
P = C(0 \leq y \leq x \leq 1)
\]
\[
Q = C(0 \leq y \leq x \leq 2)
\]
\[
P \triangleright Q = C(0 \leq y \leq x)
\]
Standard widening (2/3)

Problem: None of the representations is canonical

\[ P = \mathcal{C}(y = 0, \ 0 \leq x \leq 1) \]
Standard widening (2/3)

Problem: None of the representations is canonical

\[ P = C(y = 0, \ 0 \leq x \leq 1) \]

\[ Q = C(0 \leq y \leq x \leq 2) \]
Standard widening (2/3)

Problem: None of the representations is canonical

\[ P = \mathcal{C}(y = 0, \ 0 \leq x \leq 1) \]

\[ Q = \mathcal{C}(0 \leq y \leq x \leq 2) \]

\[ P \triangledown Q = \mathcal{C}(0 \leq y, \ 0 \leq x) \]
Standard widening (2/3)

Problem: None of the representations is canonical

\[ P = C(y = 0, 0 \leq x \leq 1) \]
\[ C(0 \leq y \leq x \leq 1, y \leq 0) \]
\[ Q = C(0 \leq y \leq x \leq 2) \]
\[ P \nabla Q = C(0 \leq y, 0 \leq x) \]
Standard widening (2/3)

Problem: None of the representations is canonical

\[ P = C(y = 0, 0 \leq x \leq 1) \]
\[ C(0 \leq y \leq x \leq 1, y \leq 0) \]
\[ Q = C(0 \leq y \leq x \leq 2) \]
\[ P \triangledown Q = C(0 \leq y, 0 \leq x) \]
\[ C(0 \leq y \leq x) \]
Standard widening (3/3)

Solution [Halbwachs, Thesis 1979]: keep for $P \triangledown Q$ 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 $P$.

$$P = C(y = 0, 0 \leq x \leq 1)$$
Standard widening (3/3)

Solution [Halbwachs, Thesis 1979]: keep for $P \sqcup Q$ 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 $P$.

\[
P = \mathcal{C}(y = 0, 0 \leq x \leq 1)
\]

\[
Q = \mathcal{C}(0 \leq y \leq x \leq 2)
\]
Standard widening (3/3)

Solution [Halbwachs, Thesis 1979]: keep for $P \cap Q$ 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 $P$.

\[ P = C(y = 0, 0 \leq x \leq 1) \]
\[ Q = C(0 \leq y \leq x \leq 2) \]

0 ≤ y already in $P$

y ≤ x mut. red. with 0 ≤ x in $P$
Solution [Halbwachs, Thesis 1979]: keep for $P \triangleleft Q$ the constraints of $Q$ which are mutually redundant with constraints of $P$ (i.e., can replace some constraints of $P$ without changing it) if they are saturated by the same generators of $P$.

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

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

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
Improving the precision

Delaying the widening (1/2)

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

Instead of computing

\[ X_0 = \perp, \quad X_1 = X_0 \nabla F(X_0), \quad X_2 = X_1 \nabla F(X_1) \ldots X_{n+1} = X_n \nabla F(X_n) \]

fix \( k > 0 \) and compute

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

or, more generally, apply the widening sporadically but infinitely often.
Delaying the widening (2/2)

Delay, loop unrolling

N. Halbwachs (Verimag/CNRS)
Delaying the widening (2/2)

Delay, loop unrolling

Easy, but often expensive.

The suitable number of delays or unrolling may depend on the program.

N. Halbwachs (Verimag/CNRS)
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 precision

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:

$$P \nabla Q = \begin{cases} 
    P \sqcup Q & \text{if dim}(Q) > \text{dim}(P) \\
    & \text{or codim}(Q) > \text{codim}(P) \\
    & \text{or } \#C_P > \#C_{P \sqcup Q} \\
    & \text{or } \#V_P > \#V_{P \sqcup Q} \\
    & \text{or } \ldots \\
    P \nabla_S Q & \text{otherwise}
\end{cases}$$

- 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 $\nabla_1$ will be better that the one provided by $\nabla_2$.
- 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 \(X'_n = X_n \land (k = n)\)
- Express \(X'_n\) as a function of \(k\), and let \(k\) tend to \(\infty\).

More precisely,

- Compute \(X'_1 = (X_0 \land k = 0) \uplus (X_1 \land k = 1)\).
- Forget the constraint \(k \leq 1\) in \(X'_1\)
- Eliminate \(k\) by existential quantification.
Improving the widening operator (4/4)

Example:

![Graph showing a line segment on a coordinate plane with axes labeled x and y.]
Example:
Example:
Improving the widening operator (4/4)

Example:
Example:
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)

\[
\begin{aligned}
\text{y} & \quad \text{x} \\
\end{aligned}
\]

\[
\begin{aligned}
\text{k} & \quad \text{x} \\
\end{aligned}
\]
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.
The widening $X \triangledown F(X)$ 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 $\nabla^L_C$ defined by:

$$P \nabla^L_C Q = (P \nabla Q) \cap C(\{c \in C \mid P \models c \land Q \models c\})$$

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 $\nabla^L_C$ defined by:

$$P \nabla^L_C Q = (P \nabla Q) \cap C(\{c \in C \mid P \models c \land Q \models c\})$$

is a widening

Choice of limiting constraints:

\[ b(c) \in C \]
Limited widening (1/2)

Example: speedometer

\[ t := d := s := 0 \]

\[ t := t + 1; s := 0 \]

\[ d := d + 1; s := s + 1 \]

\[ \text{second?} \]

\[ \text{meter and } s \leq 3? \]

\[ \text{meter and } s > 3? \]

Limit the widening with \([s := s + 1](s \leq 3) = (s \leq 4)\).
Limited widening (1/2)

Example: speedometer

\[
\begin{align*}
    t &:= t + 1; s := 0 \\
    d &:= d + 1; s := s + 1
\end{align*}
\]

second?

meter and \( s \leq 3 \)?

meter and \( s > 3 \)\

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.
Taking the program into account

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.

```plaintext
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
```

N. Halbwachs (Verimag/CNRS)

On the design of widening operators
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.

```plaintext
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
```

N. Halbwachs (Verimag/CNRS)
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.

\[
i := j := 0 := k := 0;
while i' \leq 100 do
\quad i := i+2; j := j+1;
\quad if ? then k := k+1;
\quad if j = 10 then
\quad \quad j := 0; k:= k+1;
end
\]
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.

```plaintext
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
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.

```plaintext
i := j := 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
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)

\[ i := j := 0 := k := 0; \]
\[ \text{while } i' \leq 100 \text{ do} \]
\[ \quad i := i+2; j := j+1; \]
\[ \quad \text{if } ? \text{ then } k := k+1; \]
\[ \quad \text{if } j = 10 \text{ then} \]
\[ \quad \quad j := 0; k := k+1; \]
\[ \text{end} \]
New control path (3/3)

\[
i := j := 0 := k := 0; \\
\text{while } i' \leq 100 \text{ do} \\
\quad i := i+2; j := j+1; \\
\quad \text{if } ? \text{ then } k := k+1; \\
\quad \text{if } j = 10 \text{ then} \\
\quad \quad j := 0; k := k+1; \\
\text{end}
\]
New control path (3/3)

\[
i := j := 0 := k := 0;
\]
\[
\text{while } i' \leq 100 \text{ do}
\]
\[
i := i+2; j := j+1;
\]
\[
\text{if } ? \text{ then } k := k+1;
\]
\[
\text{if } j = 10 \text{ then}
\]
\[
j := 0; k:= k+1;
\]
\[
\text{end}
\]
New control path (3/3)

```plaintext
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)

\[
i := j := 0 := k := 0;
\text{while } i' \leq 100 \text{ do}
    \quad i := i+2; j := j+1;
    \quad \text{if } ? \text{ then } k := k+1;
    \quad \text{if } j = 10 \text{ then}
        \quad j := 0; k := k+1;
\text{end}
\]
New control path (3/3)

\[
i := j := 0 := k := 0; \\
\text{while } i' \leq 100 \text{ do} \\
\quad i := i+2; j := j+1; \\
\quad \text{if } ? \text{ then } k := k+1; \\
\quad \text{if } j = 10 \text{ then} \\
\quad \quad j := 0; k := k+1; \\
\text{end}
\]

\[0 \leq k \leq \frac{11}{20} i\]
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]

\[ \tau \text{ being a relation in } \mathbb{N}^n \times \mathbb{N}^n, \text{ in which cases can we compute exactly } \tau^* = \bigcup_{k \in \mathbb{N}} \tau^k \text{ or } \tau^*(X_0)? \]
Loop acceleration (2/3)

- Restrictions on $\tau$ (e.g., $\ell(X) \land X' = X + B$, or $\ell(X) \land X' \# X + B$, $\# \in \{\leq, =, \geq\}$), flat automata (without nested loops), . . .
  \[\rightarrow\] 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)

\[ \ell?X := X + B \]
Loop acceleration (3/3)

\[ \ell?X := X + B \]

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

is a Presburger formula
Loop acceleration (3/3)

\[
\ell' \Delta X := X + B' \\
\ell \Delta X := X + B
\]

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

is a Presburger formula

Not true for interleaving of transitions
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
Abstract acceleration on polyhedra [Gonnord-Halbwachs06]

Example: the speedometer in one abstract acceleration:

\[
\begin{align*}
    t &:= t + 1; s := 0 \\
    d &:= d + 1; s := s + 1 \\
    \text{second?} \quad s &\leq 3 \\
    \text{meter and } s &> 3
\end{align*}
\]
Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

\[ \text{t} := \text{d} := \text{s} := 0; \]

while true do
    \[ \text{s} \leq 3 : \text{s}++; \text{d}++; \]
    \[ \text{true: t++; s:=0} \]
end
Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

\[
t := d := s := 0; \\
\text{while true do} \\
\quad s \leq 3 : s++; d++ \\
\quad \text{true: } t++; s := 0 \\
\text{end}
\]
The speedometer in one abstract acceleration:

\[ \text{t:=d:=s:=0; while true do s\leq3 : s++; d++; } \]
\[ \text{true: t++; s:=0 end} \]
Exact abstract computations (3/3)

The speedometer in one abstract acceleration:

t:=d:=s:=0;
while true do
    s \leq 3 : s++; d++
    true: t++; s:=0
end
The speedometer in one abstract acceleration:

```plaintext
t:=d:=s:=0;
while true do
    s <= 3 : s++; d++
    true: t++; s:=0
end
```

P = (\{(0,0,0)\} ↗ \{(1,0,0), (0,1,1), (1,4,0)\}) ∩ (s ≤ 4)

N. Halbwachs (Verimag/CNRS)
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

\[ P = (\{ (0,0,0) \} \uparrow \{ (1,0,0), (0,1,1), (1,4,0) \}) \cap (s ≤ 4) = (t ≥ 0, 0 ≤ s ≤ 4, 0 ≤ d ≤ 4 t+s) \]
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

\[
P = (\{(0,0,0)\} \uparrow \{(1,0,0),(0,1,1),(1,4,0)\}) \cap (s \leq 4) \\
= (t \geq 0, 0 \leq s \leq 4, 0 \leq d \leq 4t + s)
\]
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.