Inferring Sufficient Conditions with Backward Polyhedral Under-Approximations

Antoine Miné

CNRS & École normale supérieure
Paris, France

Workshop on Numerical and Symbolic Abstract Domains
10 September 2012
Deauville

Introduction

Inferring invariants is a well-studied problem,
with many applications
(proving correctness, optimising, ...)

Favorite method: abstract interpretation using abstract domains
(scalable, terminates thanks to $\triangledown$, modular, flexible)

> We want to infer sufficient conditions
> for a given property to hold on a given program...
> ...and still use abstractions

Focus on numeric properties
$\Longrightarrow$ numeric abstract domains (polyhedra)

Outline

- sufficient conditions
  - transition systems
  - programs
  - applications

- design effective abstract under-approximations
  - general algebraic properties
  - **polyhedral operators**
    (tests, assignments, loops, non-linear expressions)

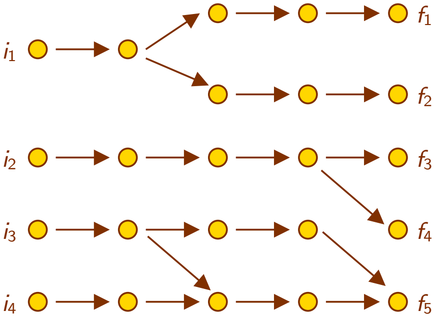
Disclamer

Work in progress, much research to be done

My goal: convince you that it might be **interesting**

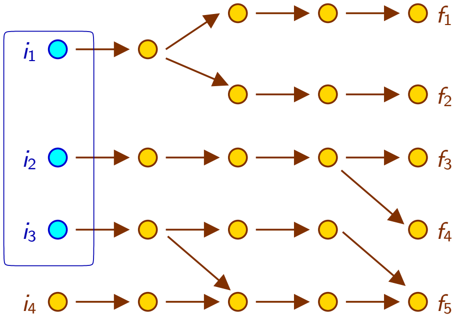
Sufficient Conditions

Transition Systems



- Σ: set of states
- $\tau \subseteq \Sigma \times \Sigma$: transition relation, possibly non-deterministic
- initial states $i_1, i_2, \ldots$ and final states $f_1, f_2, \ldots$

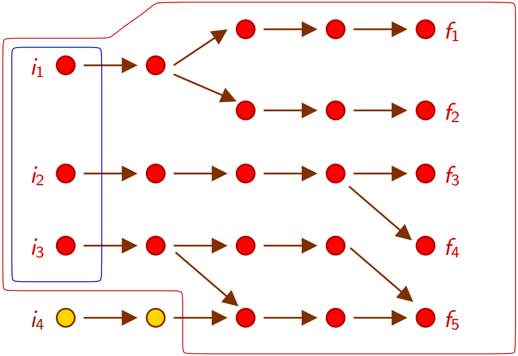
Transition Systems: Invariants



Given a set I of initial states,

compute the set $\mathrm{inv}(I)$ of reachable states

Transition Systems: Invariants

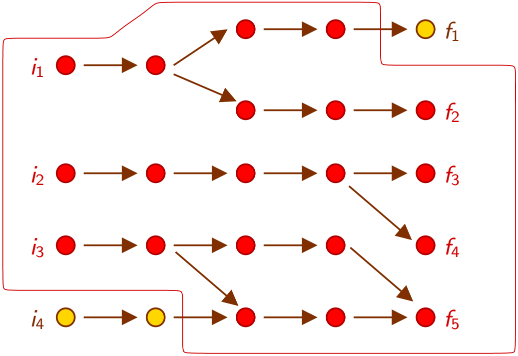


$\mathrm{inv}(I) = \mathrm{lfp}_I \; \lambda X.X \cup \mathrm{post}(X)$

where $\mathrm{post}(X) \stackrel{\text{def}}{=} \{ \sigma \in \Sigma \mid \exists \sigma' \in X \colon (\sigma', \sigma) \in \tau \}$

Smallest set containing I and invariant by τ

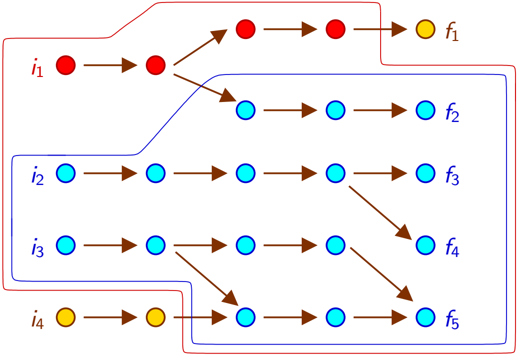
Transition Systems: Sufficient Conditions



Given a target invariant set T,

compute the largest set I such that $\mathrm{inv}(I) \subseteq T$

Transition Systems: Sufficient Conditions



$\text{cond}(T) = \text{gfp}_T\, \lambda X.X \cap \widetilde{\text{pre}}(X)$

where $\widetilde{\text{pre}}(X) \stackrel{\text{def}}{=} \{\, \sigma \in \Sigma \mid \forall \sigma' \in \Sigma: (\sigma, \sigma') \in \tau \Longrightarrow \sigma' \in X \,\}$

$\Longrightarrow$ $\text{cond}(T)$ is the largest set I such that $\text{inv}(I) \subseteq T$

Example Program Analysis

Simple (non-deterministic) loop example

```
j := [0;10]
for (i = 0; i < 100; i++) {
  j = j + [0;1];
}
assert (j <= 105);
```

Example Program Analysis

Simple (non-deterministic) loop example

```
j := [0;10]
for (i = 0; i < 100; i++) {
  j = j + [0;1];
}
assert (j <= 105);
```

Forward invariant inference:

final values of j

$\Longrightarrow j \in [0;110]$, the assertion can be violated

Example Program Analysis

Simple (non-deterministic) loop example

```
j := [0;10]
for (i = 0; i < 100; i++) {
  j = j + [0;1];
}
assert (j <= 105);
```

Classic backward analysis:

initial values of j that may lead to correct program termination

$\Longrightarrow j \in [0;10]$ (we can always choose $0 \in [0;1]$)

Example Program Analysis

Simple (non-deterministic) loop example

```
j := [0;10]
for (i = 0; i < 100; i++) {
  j = j + [0;1];
}
assert (j <= 105);
```

Backward sufficient condition analysis:

initial values of j that always lead to correct program termination

$\Longrightarrow j \in [0;5]$ (works even if we choose always $1 \in [0;1]$)

Analysis Specificities

- backwards
- handling of non-determinism
- under-approximation

Analysis Specificities

- backwards
- handling of non-determinism

$\widetilde{\mathrm{pre}}(X) \neq \mathrm{pre}(X) \stackrel{\text{def}}{=} \{\, \sigma \in \Sigma \mid \exists \sigma' \in X \colon (\sigma, \sigma') \in \tau \,\}$

we could use $\widetilde{\mathrm{pre}}(X) = \neg(\mathrm{pre}(\neg X))$
but abstract domains are seldom closed under $\neg$

$\Longrightarrow$ classic backward operators cannot be recycled easily

- under-approximation

Analysis Specificities

- backwards
- handling of non-determinism
- under-approximation

  $I' \subseteq \mathrm{cond}(T) \Longrightarrow \mathrm{inv}(I') \subseteq T$
  $\longrightarrow$ soundness requires under-approximations

  abstract domains seldom have best under-approximations
  (vs. best over-approximations via Galois connection)

  we could use an abstract X to represent $\neg X$
  $\longrightarrow$ turns over-approximations into under-approximations
  but $\neg X$ seldom represents an interesting property

  $\Longrightarrow$ classic backward operators cannot be recycled easily

Analysis Specificities

- backwards
- handling of non-determinism
- under-approximation

$\Longrightarrow$ design new, non-optimal operators on existing domains

Applications

- sufficient condition for correctness
  application to contract inference
- run-time check hoisting
- counter-example inference

Applications

- sufficient condition for correctness
- run-time check hoisting

original

```
for (i=0; i<n; i++)
    assert (i≥0 && i<N);
    t[i] = 1;
```

→

optimized

```
if (n≤N)
    for (i=0; i<n; i++)
        t[i] = 1;
else
    original program
```

- counter-example inference

Applications

- sufficient condition for correctness
- run-time check hoisting
- counter-example inference
  - sufficient initial conditions for the assertion to fail
    $\longrightarrow$ under-approximate $\mathrm{cond}(T)$ as before
  - proof that the assertion is eventually reached
    liveness property!
    $\longrightarrow$ instrument with a decreasing counter
    (idea from Cousot Cousot POPL'12)

Backward Polyhedral Under-Approximations

Backward Versions

Programs are decomposed into atomic instructions i
with well-known concrete transfer functions $\tau\{\!|\, i \,|\!\}$

$\Longrightarrow$ derive concrete backwards transfer functions $\overleftarrow{\tau}\{\!|\, i \,|\!\}$
from concrete forward ones $\tau\{\!|\, i \,|\!\}$

Backward version of $f : \mathcal{P}(X) \rightarrow \mathcal{P}(Y)$

$\overleftarrow{f} : \mathcal{P}(Y) \rightarrow \mathcal{P}(X)$
$\overleftarrow{f}(B) \stackrel{\text{def}}{=} \{\, a \in X \mid f(\{a\}) \subseteq B \,\}$

Core properties:

- $\widetilde{\text{pre}} = \overleftarrow{\text{post}}$
- $\overleftarrow{f}$ is monotonic and a complete $\cap-$morphism
- $\mathcal{P}(X) \underset{f}{\overset{\overleftarrow{f}}{\leftrightarrows}} \mathcal{P}(Y)$ (if f is a complete $\cup-$morphism)

Algebraic Properties

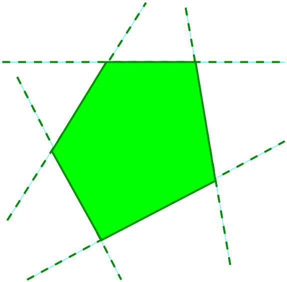
- $\overleftarrow{f \cup g} = \overleftarrow{f} \cap \overleftarrow{g}$ (but $\overleftarrow{f \cap g} \supset \overleftarrow{f} \cup \overleftarrow{g}$)
  (control-flow join)
- $\overleftarrow{f \circ g} = \overleftarrow{g} \circ \overleftarrow{f}$
  (instruction sequence)
- $f \subseteq g \iff \overleftarrow{g} \subseteq \overleftarrow{f}$
  (approximation)
- $\overleftarrow{\lambda x.\mathrm{lfp}_x\,(\lambda z.z \cup f(z))} = \lambda y.\mathrm{gfp}_y(\lambda z.z \cap \overleftarrow{f}(z))$
  (loops)

$\Longrightarrow$ break-down, abstract and combine backward functions

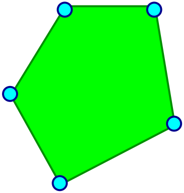
Polyhedra

domain of convex closed polyhedra

Dual representations:



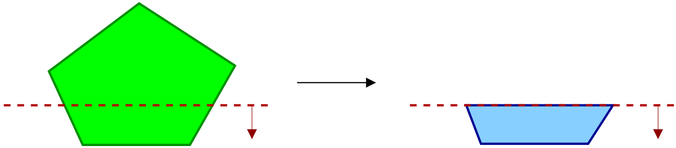
Constraints



Generators

(Cousot & Halbwachs, POPL 1978)

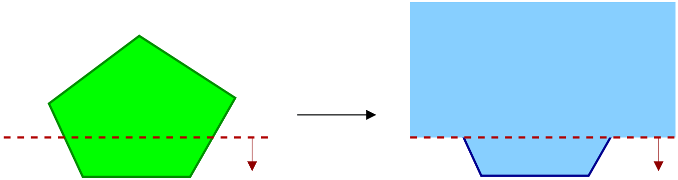
Modeling Affine Tests



Concrete Semantics:

we have $\tau \{\!| \mathbf{a} \cdot \mathbf{x} \leq b? |\!\} R = \{ \rho \in R \mid \mathbf{a} \cdot \rho(\mathbf{x}) \leq b \}$

Modeling Affine Tests

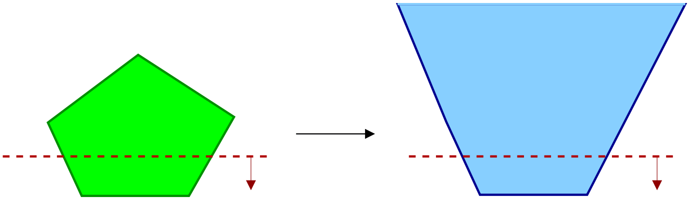


Concrete Semantics:

we have $\tau\{\!|\, \mathbf{a}\cdot\mathbf{x} \leq b? \,|\!\}\, R = \{\, \rho \in R \mid \mathbf{a}\cdot\rho(\mathbf{x}) \leq b \,\}$

and so $\overleftarrow{\tau}\{\!|\, \mathbf{a}\cdot\mathbf{x} \leq b? \,|\!\}\, R = R \cup \{\, \rho \mid \mathbf{a}\cdot\rho(\mathbf{x}) > b \,\}$

Modeling Affine Tests



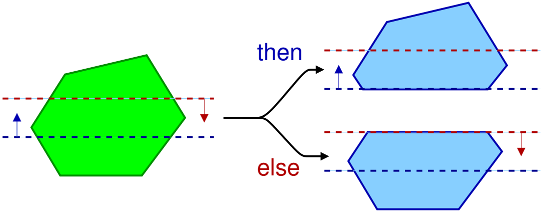
Abstract Polyhedral Semantics: $\overleftarrow{\tau}\{\!|\, \mathbf{a}\cdot\mathbf{x} \leq b?\,|\!\}^\sharp\, P$

- remove $\mathbf{a}\cdot\mathbf{x} \leq b$ from the constraint set
- remove all constraints redundant with $\mathbf{a}\cdot\mathbf{x} \leq b$

$\Longrightarrow$ under-approximation, not optimal

Note: $\lambda P.P$ always under-approximates $\overleftarrow{\tau}\{\!|\, e?\,|\!\}$

Modeling If-Then-Else



Forward semantics: $y + [0;1] \geq 0$

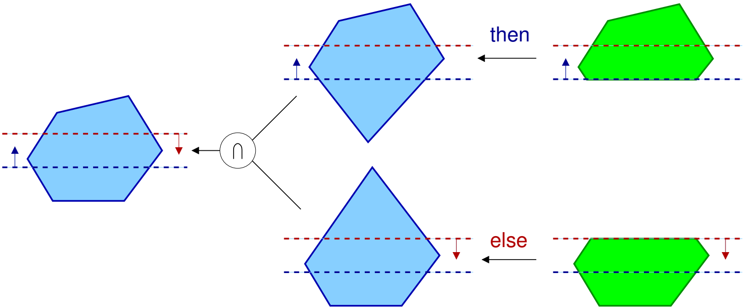
$\tau\{\!|\ \mathbf{if}\ (y + [0;1] \geq 0)\ \{\, t \,\}\ \mathbf{else}\ \{\, e \,\}\ |\!\} =$

$(\tau\{\!|\, t \,|\!\} \circ \tau\{\!|\, y + [0;1] \geq 0 \,|\!\}) \cup (\tau\{\!|\, e \,|\!\} \circ \tau\{\!|\, y + [0;1] < 0 \,|\!\})$

where $\tau\{\!|\, y + [0;1] \geq 0 \,|\!\} R = \{\, (x,y) \in R \mid y \geq -1 \,\}$

$\tau\{\!|\, y + [0;1] < 0 \,|\!\} R = \{\, (x,y) \in R \mid y < 0 \,\}$

Modeling If-Then-Else



Backward semantics:

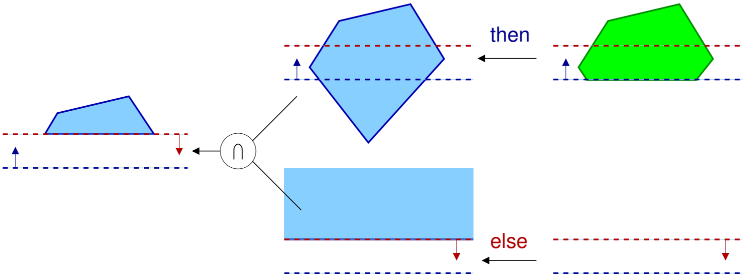
$$\overleftarrow{\tau}\{\!|\ \textbf{if}\ \ (y + [0;1] \geq 0)\ \ \{\, t \,\}\ \ \textbf{else}\ \ \{\, e \,\}\ |\!\}\ =$$

$$(\overleftarrow{\tau}\{\!|\, y + [0;1] \geq 0 \,|\!\} \circ \overleftarrow{\tau}\{\!|\, t \,|\!\}) \cap \overleftarrow{\tau}\{\!|\, y + [0;1] < 0 \,|\!\} \circ \overleftarrow{\tau}\{\!|\, e \,|\!\})$$

where $\overleftarrow{\tau}\{\!|\, y + [0;1] \geq 0 \,|\!\}\, R = R \cup \{\, (x, y) \mid y < -1 \,\}$

$\overleftarrow{\tau}\{\!|\, y + [0;1] < 0 \,|\!\}\, R = R \cup \{\, (x, y) \mid y \geq 0 \,\}$

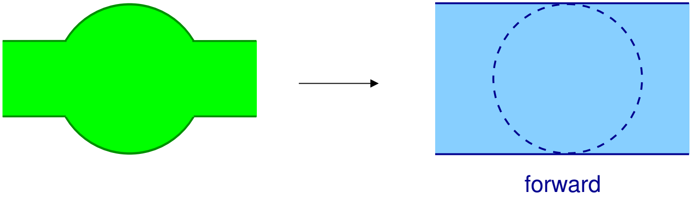
Modeling If-Then-Else



$\overleftarrow{\tau}\{\!| \cdot |\!\}$ is not strict
$\Longrightarrow$ we can recover from a coarse under-approximation, even $\emptyset$

Example: the analysis finds the else branch dead
but continues nevertheless

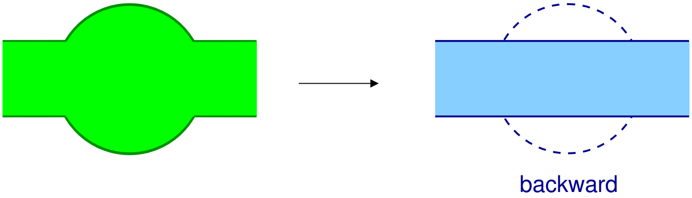
Modeling Projections



Concrete Semantics:

we have $\tau \{\!|\, x := ?\, |\!\}\, R = \{\, (x, y) \mid \exists v : (v, y) \in R \,\}$

Modeling Projections

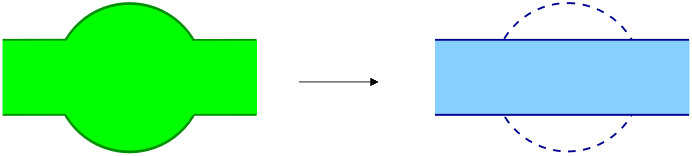


Concrete Semantics:

we have $\tau\{\!|\, x :=?\, |\!\}\, R = \{\, (x, y) \mid \exists v \colon (v, y) \in R \,\}$

and so $\overleftarrow{\tau}\{\!|\, x :=?\, |\!\}\, R = \{\, (x, y) \mid \forall v \colon (v, y) \in R \,\}$

Modeling Projections



Theorem

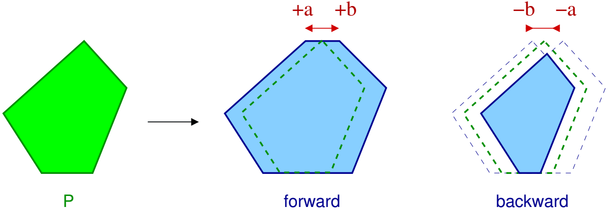
If R is convex closed, then $\overleftarrow{\tau}\{\!|\, x := ?\,|\!\}\, R$ is either R or $\emptyset$

Abstract Polyhedral Semantics:

$$\overleftarrow{\tau}\{\!|\, x := ?\,|\!\}^\sharp\, P = \begin{cases} P & \text{if } \tau\,\{\!|\, x := ?\,|\!\}^\sharp\, P = P \\ \emptyset & \text{otherwise} \end{cases} \qquad \text{(exact)}$$

Modeling Assignments

Example: $x := x + [a; b]$ (exact)



$$\tau\{\!|\, x := x + [a; b] \,|\!\}\, R \;=\; \{\, (x+v, y) \mid (x,y) \in R,\; v \in [a; b] \,\}$$

$$\overleftarrow{\tau}\{\!|\, x := x + [a; b] \,|\!\}\, R \;=\; \{\, (x, y) \mid \forall v \in [a; b]: (x+v, y) \in R \,\}$$

$$\overleftarrow{\tau}\{\!|\, x := x + [a; b] \,|\!\}^{\sharp}\, P = \tau\{\!|\, x := x - a \,|\!\}^{\sharp}\, P \cap^{\sharp} \tau\{\!|\, x := x - b \,|\!\}^{\sharp}\, P$$

General case:

- $\tau\{\!|\, x := e \,|\!\}$ can be synthesized using tests and projections $\Longrightarrow$ so can $\overleftarrow{\tau}\{\!|\, x := e \,|\!\}$ and $\overleftarrow{\tau}\{\!|\, x := e \,|\!\}^{\sharp}$
- $\tau\{\!|\, x :=? \,|\!\}$ over-approximates $\tau\{\!|\, x := e \,|\!\}$, so $\overleftarrow{\tau}\{\!|\, x :=? \,|\!\}$ under-approximates $\overleftarrow{\tau}\{\!|\, x := e \,|\!\}$ $\Longrightarrow$ use $\overleftarrow{\tau}\{\!|\, x :=? \,|\!\}^{\sharp}$

Loops

Concrete Semantics:

$$\tau\{\!|\, \textbf{while}\ (e)\ \{\, b \,\}\, |\!\}\, X \stackrel{\text{def}}{=} \tau\{\!|\, \neg e?\, |\!\}\, (\mathrm{lfp}_X\ \lambda Y.Y \cup (\tau\{\!|\, b\, |\!\} \circ \tau\{\!|\, e?\, |\!\})Y)$$

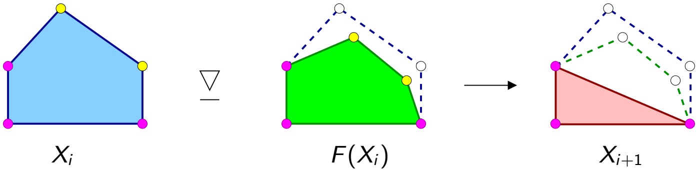
$$\overleftarrow{\tau}\{\!|\, \textbf{while}\ (e)\ \{\, b \,\}\, |\!\}\, X = \mathrm{gfp}_{(\overleftarrow{\tau}\{\!|\, \neg e?\, |\!\} X)}\ \lambda Y.Y \cap (\overleftarrow{\tau}\{\!|\, e?\, |\!\} \circ \overleftarrow{\tau}\{\!|\, b\, |\!\})Y$$

Forward Abstract Semantics:

abstract $\mathrm{lfp}_X\, F$ as the limit of $\begin{cases} X_0^\sharp \stackrel{\text{def}}{=} X^\sharp \\ X_{i+1}^\sharp \stackrel{\text{def}}{=} X_i^\sharp \triangledown F^\sharp(X_i^\sharp) \end{cases}$ where

- $X^\sharp$ and $F^\sharp$ over-approximate X and F
- the widening $\triangledown$ ensures convergence (e.g., remove unstable constraints)

Lower Widening

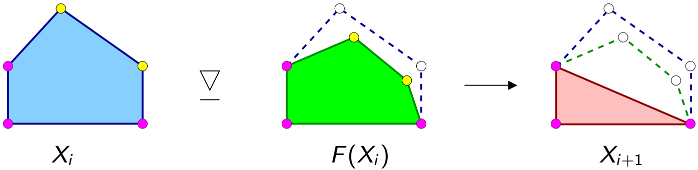


Backward Abstract Semantics:

abstract $\mathrm{gfp}_X\ F$ as the limit of $\begin{cases} X_0^\sharp \stackrel{\text{def}}{=} X^\sharp \\ X_{i+1}^\sharp \stackrel{\text{def}}{=} X_i^\sharp\ \underline{\nabla}\ F^\sharp(X_i^\sharp) \end{cases}$ where

- $X^\sharp$ and $F^\sharp$ under-approximate X and F
- $\underline{\nabla}$ is a **lower widening**
  - $A^\sharp\ \underline{\nabla}\ B^\sharp$ under-approximates $A^\sharp \cap B^\sharp$
  - $\forall (X_n^\sharp)_{n\in\mathbb{N}}$: the sequence $Y_0^\sharp \stackrel{\text{def}}{=} X_0^\sharp$, $Y_{i+1}^\sharp \stackrel{\text{def}}{=} Y_i^\sharp\ \underline{\nabla}\ X_{i+1}^\sharp$ stabilizes in finite time: $\exists i$: $Y_{i+1}^\sharp = Y_i^\sharp$

Lower Widening



Example Polyhedral Lower Widening:

- remove unstable generators
- optionally use thresholds

$\underline{\nabla}$ introduced formally by Cousot, along ∇
but no instance until now

$\underline{\nabla}$ is not a narrowing!

Expression Abstraction

Forward Expression Abstraction:

Idea: replace $\tau\,\{\!|\, x := e \,|\!\}\, R$ with $\tau\,\{\!|\, x := f \,|\!\}\, R$ when

- $x := f$ is simpler to abstract than $x := e$
- $\forall \rho \in R$: $[\![\, e \,]\!]\, \rho \subseteq [\![\, f \,]\!] \rho$ (soundness)

Backward Expression Abstraction:

Theorem

$\forall X$: $\overleftarrow{\tau}\,\{\!|\, x := e \,|\!\}\, X \supseteq (\overleftarrow{\tau}\,\{\!|\, v := f \,|\!\}\, X) \cap R$

$\Longrightarrow$ replace $\overleftarrow{\tau}\,\{\!|\, x := e \,|\!\}^\sharp\, X^\sharp$ with $(\overleftarrow{\tau}\,\{\!|\, x := f \,|\!\}^\sharp\, X^\sharp) \cap^\sharp R^\sharp$

e.g.: $\overleftarrow{\tau}\,\{\!|\, x := y * z \,|\!\}^\sharp\, P \longrightarrow (\overleftarrow{\tau}\,\{\!|\, x := [0;1] * z \,|\!\}^\sharp\, P) \cap Q$
if $Q \Rightarrow y \in [0;1]$

over-approximating e under-approximates $\overleftarrow{\tau}\{\!|\, x := e \,|\!\}$*!*

(also works on tests)

Prototype

extremely simple proof-of-concept

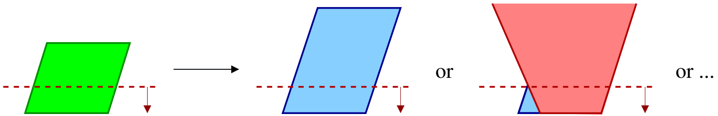
- analyzes a toy-language
- forward invariant + backward sufficient condition analysis
- abstract interpretation by induction on the syntax
- polyhedral abstractions based on Apron
- on-line analysis: `http://www.di.ens.fr/~mine/banal`
- sources freely available (in Ocaml)

Example analyses:

- simple loops (`for (...)  j = j + [0;1]`)
- bubble sort (Cousot Halbwachs 78)

Short-Comings

- Abstraction of tests



$\Longrightarrow$ how to choose now to maximize the end result?

- Lower widening

$\Longrightarrow$ widening is the usual suspect

- Experiments

Related Work

- $w(l)p$ calculus (Dijkstra 75, Morris 97)
- (finite) model-checking (Dams 96)
- over-approximating backward abstract interpretation (Bourdoncle 93, Rival 05)
- under-approximations
  - exact disjunctive completions
  - path enumeration (Moy 08)
  - domains closed by complement (Lev-Ami et al. 07)
- higher-order abstract interpretation
  - abstract lower closure operators (Massé 02)
  - under-approximation on powersets (Schmidt 04)

Conclusion

It **seems possible** to:

- infer sound sufficient conditions
- for non-deterministic, infinite-state programs
- using non-trivial under-approximations
- on infinite-state abstract domains

Our contribution:

1. general properties for compositional analysis design
2. example abstract transfer functions on polyhedra

Much more work to do to make it practical!