Abstract Interpretation

Thomas Jensen

INRIA

Ecole Jeunes Chercheurs en Programmation
May 2013
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time, we make an union with the previous property.
- We "execute" the program until stability:
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We “execute” the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;
{ (0,0) }
while (x<6) {
  if (?) {
    { }
y = y+2;
    { }
  };
  { 
x = x+1;
    { }
  }
}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We "execute" the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```
x = 0; y = 0;

{(0,0)
}

while (x<6) {
    if (?) {
        { (0,0)
            y = y+2;
            {
            }
        }
        {
            y = y+2;
            {
            }
        }
    }
    {
        x = x+1;
        {
        }
    }

```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We "execute" the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;
((0,0))
while (x<6) {
  if (?) {
    ((0,0))
    y = y+2;
    (0,2)
  };
    {
    k
    x = x+1;
    {
    }
  }
}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We "execute" the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;
{(0,0)}
while (x<6) {
  if (?) {
    {(0,0)}
    y = y+2;
    {(0,2)}
  };
  {(0,0), (0,2)}
x = x+1;
  }
}
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We “execute” the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;

{(0,0)}

while (x<6) {
  if (?) {
    {(0,0)}
    y = y+2;
    {(0,2)}
  };
    {(0,0), (0,2)}
  x = x+1;
    {(1,0), (1,2)}
}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We "execute" the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;
    { (0,0), (1,0), (1,2) }
while (x<6) {
    if (?) {
        { (0,0) }
        y = y+2;
        { (0,2) }
    };
        { (0,0), (0,2) }
        x = x+1;
        { (1,0), (1,2) }
    }
```

Static program analysis
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We “execute” the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;  
{ (0,0), (1,0), (1,2) }  
while (x<6) {  
  if (?) {  
    { (0,0), (1,0), (1,2) }  
    y = y+2;  
    { (0,2) }  
  }  
  { (0,0), (0,2) }  
  x = x+1;  
  { (1,0), (1,2) }  
}  
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We "execute" the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```c
x = 0; y = 0;
{ (0,0), (1,0), (1,2) } 
while (x<6) {
  if (?) {
    { (0,0), (1,0), (1,2) } 
    y = y+2;
    { (0,2), (1,2), (1,4) } 
  };
  { (0,0), (0,2) } 
  x = x+1;
  { (1,0), (1,2) } 
} 
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We "execute" the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;

{(0,0), (1,0), (1,2)   }

while (x<6) {
  if (?) {
    {(0,0), (1,0), (1,2)   }
    y = y+2;
    {(0,2), (1,2), (1,4)   }
  };

  {(0,0), (0,2), (1,0), (1,2), (1,4)   }

  x = x+1;
  {(1,0), (1,2)   }

}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

▶ A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
▶ When a point is reached for a second time we make an union with the previous property.
▶ We "execute" the program until stability
  ▶ It may take an infinite number of steps...
  ▶ But the limit always exists (explained later)

```java
x = 0; y = 0;
{ (0,0), (1,0), (1,2) }
while (x<6) {
  if (?) {
    { (0,0), (1,0), (1,2) }
    y = y+2;
    { (0,2), (1,2), (1,4) }
  };
  { (0,0), (0,2), (1,0), (1,2), (1,4) }
  x = x+1;
  { (1,0), (1,2), (2,0), (2,2), (2,4) }
}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Collecting semantics

- A state property is a subset in $\mathcal{P}(\mathbb{Z}^2)$ of $(x, y)$ values.
- When a point is reached for a second time we make an union with the previous property.
- We “execute” the program until stability
  - It may take an infinite number of steps...
  - But the limit always exists (explained later)

```plaintext
x = 0; y = 0;
{(0,0), (1,0), (1,2), ...}
while (x<6) {
  if (?) {
    {(0,0), (1,0), (1,2), ...}
    y = y+2;
    {(0,2), (1,2), (1,4), ...}
  };
  {(0,0), (0,2), (1,0), (1,2), (1,4), ...}
  x = x+1;
  {(1,0), (1,2), (2,0), (2,2), (2,4), ...}
}
{(6,0), (6,2), (6,4), (6,6), ...}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x \mathrel{\leq} 0 \land y \mathrel{\leq} 0
\]
\[
C ::= < | \leq | \geq | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x C 0 \land y C 0
\]
\[
C ::= < | \leq | = | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

```latex
x = 0; y = 0;
x = 0 \land y = 0
```

```latex
while (x<6) {
    if (?) {
        x = 0 \land y = 0
        y = y+2;
    }
    x = x+1;
}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[ P ::= x C 0 \land y C 0 \]

\[ C ::= < | \leq | = | > | \geq \]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

\[ x = 0; y = 0; \]
\[ x = 0 \land y = 0 \]
\[ \text{while} \ (x < 6) \{ \]
\[ \text{if} \ (?) \{ \]
\[ x = 0 \land y = 0 \]
\[ y = y + 2; \]
\[ x = 0 \land y > 0 \text{ over-approximation!} \]
\[ \}; \]
\[ x = x + 1; \]
\[ \} \]
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[ P ::= \ x \ C \ 0 \ \land \ y \ C \ 0 \]

\[ C ::= < | \leq \ | = | > | \geq \]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

```plaintext
x = 0; y = 0;
\quad x = 0 \ \land \ y = 0

\textbf{while} (x<6) {
    \quad x = 0 \ \land \ y = 0
    \quad y = y+2;
    \quad x = 0 \ \land \ y > 0
}\;
\quad x = 0 \ \land \ y \geq 0
\quad x = x+1;
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x \leq 0 \wedge y \leq 0
\]
\[
C ::= < | \leq | = | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

\[
x = 0; y = 0; \\
x = 0 \wedge y = 0
\]
\[
\text{while } (x<6) \{
\]
\[
\text{if } (?) \{
\]
\[
x = 0 \wedge y = 0
\]
\[
y = y+2;
\]
\[
x = 0 \wedge y > 0
\]
\[
}\;
\]
\[
x = 0 \wedge y \geq 0
\]
\[
x = x+1;
\]
\[
x > 0 \wedge y \geq 0 \text{ over-approximation !}
\]
\[
\}
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x \in C 0 \wedge y \in C 0
\]

\[
C ::= < | \leq | = | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

\[
x = 0; y = 0;
\]

\[
x \geq 0 \wedge y \geq 0
\]

\[
\textbf{while} (x<6) \{ \\
\textbf{if} (?) \{
\]

\[
x = 0 \wedge y = 0
\]

\[
y = y+2;
\]

\[
x = 0 \wedge y > 0
\]

\[
\}
\]

\[
x = x+1;
\]

\[
x > 0 \wedge y \geq 0
\]

\[
\}
\]
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x \in C 0 \land y \in C 0
\]
\[
C ::= < | \leq | = | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

```plaintext
x = 0; y = 0;
  x \geq 0 \land y \geq 0

while (x<6) {
  if (?) {
    x \geq 0 \land y \geq 0
    y = y+2;
    x = 0 \land y > 0
  }
  x = x+1;
  x > 0 \land y \geq 0
}
```
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x \in C 0 \land y \in C 0
\]

\[
C ::= < | \leq | = | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

\[
x = 0; y = 0;
\]

\[
x \geq 0 \land y \geq 0
\]

\[
\text{while} \ (x < 6) \{
\]

\[
\text{if} \ (?) \{
\]

\[
x \geq 0 \land y \geq 0
\]

\[
y = y + 2;
\]

\[
x \geq 0 \land y > 0
\]

\[
\}
\]

\[
x = 0 \land y \geq 0
\]

\[
x = x + 1;
\]

\[
x > 0 \land y \geq 0
\]

\[
\}
\]
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= \ x \ C \ 0 \ \land \ y \ C \ 0
\]

\[
C ::= \ < | \leq | = | > | \geq
\]

To stay in the domain of selected properties, we over-approximate the concrete properties.

\[
x = 0; \ y = 0;
\]

\[
x \geq 0 \ \land \ y \geq 0
\]

\[
\text{while} \ (x < 6) \ {
\text{if} \ (\ ?) \ {
\ x \geq 0 \ \land \ y \geq 0
\ y = y + 2;
\ x \geq 0 \ \land \ y > 0
\};
\ x \geq 0 \ \land \ y \geq 0
\ x = x + 1;
\ x > 0 \ \land \ y \geq 0
\}
\]
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x \in \{0\} \land y \in \{0\}
\]

\[
C ::= < | \leq | = | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

\[
\begin{align*}
x &= 0; \\
y &= 0; \\
x \geq 0 \land y \geq 0
\end{align*}
\]

\[
\textbf{while} \ (x < 6) \ {\}
\]

\[
\begin{align*}
\textbf{if} \ ? \ {\}
\end{align*}
\]

\[
\begin{align*}
x &= 0 \land y \geq 0
\end{align*}
\]

\[
\begin{align*}
y &= y + 2; \\
x \geq 0 \land y > 0
\end{align*}
\]

\[
\begin{align*}
};
\end{align*}
\]

\[
\begin{align*}
x &= 0 \land y \geq 0
\end{align*}
\]

\[
\begin{align*}
x &= x + 1;
\end{align*}
\]

\[
\begin{align*}
x &> 0 \land y \geq 0
\end{align*}
\]

\[
\begin{align*}
}\}
\end{align*}
\]
A flavor of abstract interpretation

Abstract interpretation executes programs on state properties instead of states.

**Approximation**

- The set of manipulated properties may be restricted to ensure computability of the semantics.

Example: sign of variables

\[
P ::= x \leq 0 \land y \leq 0
\]

\[
C ::= < | \leq | = | > | \geq
\]

- To stay in the domain of selected properties, we over-approximate the concrete properties.

```plaintext
x = 0; y = 0;
x \geq 0 \land y \geq 0

while (x<6) {
    if (?) {
        x \geq 0 \land y \geq 0
        y = y+2;
        x \geq 0 \land y > 0
    }
    x \geq 0 \land y \geq 0
    x = x+1;
    x > 0 \land y \geq 0
} x \geq 0 \land y \geq 0
```
Another example: the interval analysis

For each point $k$ and each numeric variable $x$, we infer an interval in which $x$ must belong to.

Example: insertion sort, array access verification

```plaintext
assert(T.length=100); i=1;

while (i<T.length) {
    p = T[i]; j = i-1;
    while (0<=j and T[j]>p) {
        T[j]=T[j+1]; j = j-1;
    }
    T[j+1]=p; i = i+1;
}
```

Another example: the polyhedral analysis

For each point \( k \) and we infer invariant linear equality and inequality relationships among variables.

Example: insertion sort, array access verification

```plaintext
assert(T.length>=1); i=1;

while i<T.length {
    p = T[i]; j = i-1;
    while 0<=j and T[j]>p {
        T[j]=T[j+1]; j = j-1;
    };
    T[j+1]=p; i = i+1;
}
```

\( \{1 \leq i \leq T.length\} \)

\( \{1 \leq i \leq T.length - 1\} \)

\( \{1 \leq i \leq T.length - 1 \land -1 \leq j \leq i - 1\} \)

\( \{1 \leq i \leq T.length - 1 \land 0 \leq j \leq i - 1\} \)

\( \{1 \leq i \leq T.length - 1 \land -1 \leq j \leq i - 2\} \)

\( \{1 \leq i \leq T.length - 1 \land -1 \leq j \leq i - 1\} \)

\( \{2 \leq i \leq T.length + 1 \land -1 \leq j \leq i - 2\} \)
Collecting Semantics

We will consider a collecting semantics that give us the set of reachable states $\llbracket p \rrbracket_k^{\text{col}}$ at each program points $k$.

$$\forall k \in P, \ [p]_k^{\text{col}} = \{ \rho | (k, \rho) \in [p]\}$$

**Theorem**

$\llbracket p \rrbracket^{\text{col}}$ may be characterized as the least fixpoint of the following equation system.

$$\forall k \in \text{labels}(p), \ X_k = X_k^{\text{init}} \cup \bigcup_{(k', i, k) \in p} \llbracket i \rrbracket (X_{k'})$$

with $X_k^{\text{init}} = \begin{cases} \text{Env} & \text{if } k = k_{\text{init}} \\ \emptyset & \text{otherwise} \end{cases}$

and

$$\forall i \in \text{Instr}, \forall X \subseteq \text{Env}, \ [i] (X) = \left\{ \rho_2 | \exists \rho_1 \in X, \rho_1 \xrightarrow{i} \rho_2 \right\} = \text{post} \left[ \xrightarrow{i} \right] (X)$$
Example

For the following program, $\llbracket P \rrbracket^{\text{col}}$ is the least solution of the following equation system:

\[
\begin{align*}
X_0 &= \text{Env} \\
X_1 &= \llbracket x := ? \rrbracket (X_0) \\
X_2 &= \llbracket x < 0 \rrbracket (X_1) \cup X_4 \\
X_3 &= \llbracket x < 0 \rrbracket (X_2) \\
X_4 &= \llbracket x := x + 1 \rrbracket (X_3) \\
X_5 &= \llbracket x \geq 0 \rrbracket (X_2) \\
X_6 &= \llbracket y := x \rrbracket (X_5) \\
X_7 &= \llbracket x \geq 0 \rrbracket (X_1) \\
X_8 &= \llbracket y := 0 \rrbracket (X_7) \\
X_9 &= X_6 \cup X_8
\end{align*}
\]
Collecting semantics and exact analysis
The \((X_k)_{i=1..N}\) are hence specified as the least solution of a fixpoint equation system
\[
X_k = F_k(X_1, X_2, \ldots, X_N), \quad k \in labels(p)
\]
or, equivalently \(\vec{X} = \vec{F}(\vec{X})\).

Exact analysis:

- Thanks to Knaster-Tarski, the least solution exists (complete lattice, \(F_k\) are monotone functions),
- Kleen fixpoint theorem (\(F_k\) are continuous functions) says it is the limit of
\[
X_k^0 = \emptyset, \quad X_k^{n+1} = F_k(X_1^n, X_2^n, \ldots, X_N^n)
\]

Uncomputable problem:

- Representing the \(X_k\) may be hard (infinite sets)
- The limit may not be reachable in a finite number of steps
Approximate analysis

Exact analysis:
Least solution of $X = F(X)$ in the complete lattice $\langle \mathcal{P}(\text{Env})^N, \subseteq, \cup, \cap \rangle$
or limit of $X^0 = \bot, X^{n+1} = F(X^n)$

Approximate analysis:

- **Static approximation:** we replace the concrete lattice $\langle \mathcal{P}(\text{Env}), \subseteq, \cup, \cap \rangle$ by an abstract lattice $\langle L^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\# \rangle$
  - whose elements can be (efficiently) represented in computers,
  - in which we know how to compute $\sqcup^\#, \sqcap^\#, \sqsubseteq^\#$, …
and we “transpose” the equation $X = F(X)$ of $\mathcal{P}(\text{Env})^N$ into $(L^\#)^N$.

- **Dynamic approximation:** when $L^\#$ does not verify the ascending chain condition, the iterative computation may not terminate in a finite number of steps (or sometimes too slowly). In this case, we can only approximate the limit (see widening/narrowing).
What is a *good* approximation space?

**Definition**

Let \((L_1, \sqsubseteq_1, \sqcup_1, \sqcap_1)\) and \((L_2, \sqsubseteq_2, \sqcup_2, \sqcap_2)\) two complete lattices. A pair of functions \(\alpha \in L_1 \rightarrow L_2\) and \(\gamma \in L_2 \rightarrow L_1\) is a Galois connection if it verifies the condition

\[
\forall x_1 \in L_1, \forall x_2 \in L_2, \quad \alpha(x_1) \sqsubseteq_2 x_2 \iff x_1 \sqsubseteq_1 \gamma(x_2)
\]

A complete lattice \((A^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#)\) is a good approximation space if there exists a Galois connection between \((A, \subseteq, \cup, \cap)\) and \((A^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#)\).
Static program analysis
Conclusion: what is a good approximation space?

1. Concrete world: a complete lattice, generally of the form \((\mathcal{P}(\mathcal{D}), \subseteq, \cup, \cap)\)
2. Abstract world: a complete lattice \((A^\#, \subseteq^#, \cup^#, \cap^#)\)
3. Link between them: Galois connection.

\[
\left(\left(\mathcal{P}(\mathcal{D}), \subseteq, \cup, \cap\right) \xrightarrow{\gamma} \left(A^\#, \subseteq^#, \cup^#, \cap^#\right)\right)
\]

\(a^\# \in A^\# \) is a correct approximation of \(a \in \mathcal{P}(\mathcal{D})\)

\[
\iff \alpha(a) \subseteq^# a^#
\]

\[
\iff a \subseteq \gamma(a^#)
\]

\(\alpha\): abstraction function \(\gamma\): concretisation function

Remark: in practice, \(\gamma\) is sufficient to prove the soundness of analyses, but we lose some nice theorems...
Just put some $\#$...

From $\mathcal{P}(\text{Env})$ to $\text{Env}^\#$

control flow graph

collecting semantics

abstract semantics

\[ X_0 = \text{Env} \]
\[ X_1 = \llbracket x := ? \rrbracket (X_0) \]
\[ X_2 = \llbracket x < 0 \rrbracket (X_1) \cup X_4 \]
\[ X_3 = \llbracket x < 0 \rrbracket (X_2) \]
\[ X_4 = \llbracket x := x + 1 \rrbracket (X_3) \]
\[ X_5 = \llbracket x \geq 0 \rrbracket (X_2) \]
\[ X_6 = \llbracket y := x \rrbracket (X_5) \]
\[ X_7 = \llbracket x \geq 0 \rrbracket (X_1) \]
\[ X_8 = \llbracket y := 0 \rrbracket (X_7) \]
\[ X_9 = X_6 \cup X_8 \]

\[ X_0^\# = \top_{\text{Env}} \]
\[ X_1^\# = \llbracket x := ? \rrbracket^\# (X_0^\#) \]
\[ X_2^\# = \llbracket x < 0 \rrbracket^\# (X_1^\#) \cup^\# X_4^\# \]
\[ X_3^\# = \llbracket x < 0 \rrbracket^\# (X_2^\#) \]
\[ X_4^\# = \llbracket x := x + 1 \rrbracket^\# (X_3^\#) \]
\[ X_5^\# = \llbracket x \geq 0 \rrbracket^\# (X_2^\#) \]
\[ X_6^\# = \llbracket y := x \rrbracket^\# (X_5^\#) \]
\[ X_7^\# = \llbracket x \geq 0 \rrbracket^\# (X_1^\#) \]
\[ X_8^\# = \llbracket y := 0 \rrbracket^\# (X_7^\#) \]
\[ X_9^\# = X_6^\# \cup^\# X_8^\# \]
Abstract semantics: the ingredients

- A lattice structure \((Env^#, \sqsubseteq_{Env}, \sqcup_{Env}, \sqcap_{Env}, \bot_{Env}, \top_{Env})\)
  - \(\sqsubseteq_{Env}\) is an approximation of \(\subseteq\)
  - \(\sqcup_{Env}\) is an approximation of \(\cup\)
  - \(\sqcap_{Env}\) is an approximation of \(\cap\)
  - \(\bot_{Env}\) is an approximation of \(\emptyset\)
  - \(\top_{Env}\) is an approximation of \(Env\)

- For all \(x \in V, e \in Exp\), \([x := e]^# \in Env^# \rightarrow Env^#\) an approximation of \([x := e]\)

- For all \(t \in test\), \([t]^# \in Env^# \rightarrow Env^#\) an approximation of \([t]\).
module type Lattice =
  sig
    type t
    val order_dec : t -> t -> bool
    val join : t -> t -> t
    val meet : t -> t -> t
    val bottom : unit -> t
    val top : unit -> t
  end

module type EnvAbstraction =
  sig
    module L : Lattice
    val assign : var -> expr -> L.t -> L.t
    val backward_test : test -> L.t -> L.t
  end

module Make =
  functor (AbEnv:EnvAbstraction) ->
  struct
    (* Generic analyser *)
  end
An abstraction by signs

\[ \begin{aligned}
\bot & \text{ represents the property } \emptyset \\
- & \text{ represents the property } \{z \mid z < 0\} \\
0 & \text{ represents the property } \{0\} \\
+ & \text{ represents the property } \{z \mid z > 0\} \\
-0 & \text{ represents the property } \{z \mid z \leq 0\} \\
+0 & \text{ represents the property } \{z \mid z \geq 0\} \\
\top & \text{ represents the property } \mathbb{Z}
\end{aligned} \]

\[ Env^\# = \mathcal{V} \rightarrow \text{Sign} : \text{a sign is associated to each variable.} \]
An abstraction by signs: example

\[
\begin{align*}
X_0^\# & = T_{Env}^\# \\
X_1^\# & = \llbracket x := ? \rrbracket^\# (X_0^\#) \\
X_2^\# & = \llbracket x < 0 \rrbracket^\# (X_1^\#) \sqcup X_4^\# \\
X_3^\# & = \llbracket x < 0 \rrbracket^\# (X_2^\#) \\
X_4^\# & = \llbracket x := x + 1 \rrbracket^\# (X_3^\#) \quad \text{which simplifies into} \\
X_5^\# & = \llbracket x \geq 0 \rrbracket^\# (X_2^\#) \\
X_6^\# & = \llbracket y := x \rrbracket^\# (X_5^\#) \\
X_7^\# & = \llbracket x \geq 0 \rrbracket^\# (X_1^\#) \\
X_8^\# & = \llbracket y := 0 \rrbracket^\# (X_7^\#) \\
X_9^\# & = X_6^\# \sqcup X_8^\#
\end{align*}
\]

with

\[
\begin{align*}
succ^\#(\bot) & = \bot \\
succ^\#(-) & = -0 \\
succ^\#(0) & = succ^\#(+) = succ^\#(+0) = + \\
succ^\#(−0) & = succ^\#(T) = T
\end{align*}
\]

\[
\begin{align*}
X_0^\# & = \llbracket x : \top ; y : \top \rrbracket \\
X_1^\# & = X_0^\#[x \mapsto \top] \\
X_2^\# & = X_1^\#[x \mapsto -] \sqcup X_4^\# \\
X_3^\# & = X_2^#[x \mapsto -] \\
X_4^\# & = X_3^#[x \mapsto succ^\#(X_3^#(x))] \\
X_5^\# & = X_2^#[x \mapsto +0] \\
X_6^\# & = X_5^#[y \mapsto X_5^#(x)] \\
X_7^\# & = X_1^#[x \mapsto +0] \\
X_8^\# & = X_7^#[y \mapsto 0] \\
X_9^\# & = X_6^\# \sqcup X_8^#
\end{align*}
\]
Abstraction by intervals

\[ \text{Int} \overset{\text{def}}{=} \{ [a, b] \mid a, b \in \overline{\mathbb{Z}}, \ a \leq b \} \cup \{ \bot \} \]

with \( \overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\} \).

\( \bot \) represents \( \emptyset \) and \([a, b]\) the property \( \{z \mid a \leq z \leq b\} \).

Envo\(\#\) \overset{\text{def}}{=} \mathcal{V} \rightarrow \text{Int} : \) an interval is associated to each variable.
Abstraction by intervals : example

\[
\begin{align*}
X^\#_0 &= \top^\#_{\text{Env}} \\
X^\#_1 &= \llbracket x := ? \rrbracket^\#(X^\#_0) \\
X^\#_2 &= \llbracket x < 0 \rrbracket^\#(X^\#_1) \sqcup^\# X^\#_4 \\
X^\#_3 &= \llbracket x < 0 \rrbracket^\#(X^\#_2) \\
X^\#_4 &= \llbracket x := x + 1 \rrbracket^\#(X^\#_3) \\
X^\#_5 &= \llbracket x \geq 0 \rrbracket^\#(X^\#_2) \\
X^\#_6 &= \llbracket y := x \rrbracket^\#(X^\#_5) \\
X^\#_7 &= \llbracket x \geq 0 \rrbracket^\#(X^\#_1) \\
X^\#_8 &= \llbracket y := 0 \rrbracket^\#(X^\#_7) \\
X^\#_9 &= X^\#_6 \sqcup^\# X^\#_8
\end{align*}
\]

\[
\begin{align*}
X^\#_0 &= [x : [-\infty, +\infty]; y : [-\infty, +\infty]] \\
X^\#_1 &= X^\#_0[x \mapsto [-\infty, +\infty]] \\
X^\#_2 &= X^\#_1[x \mapsto X^\#_1(x) \sqcap^\# [-\infty, -1]] \sqcup^\# X^\#_4 \\
X^\#_3 &= X^\#_2[x \mapsto X^\#_2(x) \sqcap^\# [-\infty, -1]] \\
X^\#_4 &= X^\#_3[x \mapsto \text{succ}^\#(X^\#_3(x))] \\
X^\#_5 &= X^\#_2[x \mapsto X^\#_2(x) \sqcap^\# [0, +\infty]] \\
X^\#_6 &= X^\#_5[y \mapsto X^\#_5(x)] \\
X^\#_7 &= X^\#_1[x \mapsto X^\#_1(x) \sqcap^\# [0, +\infty]] \\
X^\#_8 &= X^\#_7[y \mapsto [0, 0]] \\
X^\#_9 &= X^\#_6 \sqcup^\# X^\#_8
\end{align*}
\]

with

\[
\begin{align*}
\text{succ}^\#(\bot) &= \bot \\
\text{succ}^\#([a, b]) &= [a + 1, b + 1]
\end{align*}
\]
A small While analyzer is available (sign and interval analysis).

http://sawd.irisa.fr/scripts/while/
Abstraction by intervals

\[ \text{Int} \overset{\text{def}}{=} \{ [a, b] \mid a, b \in \overline{\mathbb{Z}}, a \leq b \} \cup \{ \bot \} \quad \text{with} \quad \overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\} \]

Lattice:

\[
\begin{align*}
I \in \text{Int} & \quad \bot \sqsubseteq_{\text{Int}} I \\
\begin{array}{c}
c \leq a \quad b \leq d \\
a, b, c, d \in \overline{\mathbb{Z}}
\end{array} & \quad [a, b] \sqsubseteq_{\text{Int}} [c, d]
\end{align*}
\]

\[
\begin{align*}
I \sqcup_{\text{Int}} \bot & \quad \overset{\text{def}}{=} I, \ \forall I \in \text{Int} \\
\bot \sqcup_{\text{Int}} I & \quad \overset{\text{def}}{=} I, \ \forall I \in \text{Int} \\
[a, b] \sqcup_{\text{Int}} [c, d] & \quad \overset{\text{def}}{=} [\min(a, c), \max(b, d)]
\end{align*}
\]

\[
\begin{align*}
I \sqcap_{\text{Int}} \bot & \quad \overset{\text{def}}{=} \bot, \ \forall I \in \text{Int} \\
\bot \sqcap_{\text{Int}} I & \quad \overset{\text{def}}{=} \bot, \ \forall I \in \text{Int} \\
[a, b] \sqcap_{\text{Int}} [c, d] & \quad \overset{\text{def}}{=} \rho_{\text{Int}}([\max(a, c), \min(b, d)])
\end{align*}
\]
with \( \rho_{\text{Int}} \in (\mathbb{Z} \times \mathbb{Z}) \to \text{Int} \) defined by

\[
\rho_{\text{Int}}(a, b) = \begin{cases} 
[a, b] & \text{if } a \leq b, \\
\bot & \text{otherwise}
\end{cases}
\]

\( \bot_{\text{Int}} \overset{\text{def}}{=} \bot \)

\( \top_{\text{Int}} \overset{\text{def}}{=} [-\infty, +\infty] \)

\[
\alpha_{\text{Int}}(S) \overset{\text{def}}{=} \bot \quad \text{if } S = \emptyset
\]

\[
\alpha_{\text{Int}}(S) \overset{\text{def}}{=} [\min(S), \max(S)] \quad \text{otherwise}
\]

\[
\gamma_{\text{Int}}(\bot) \overset{\text{def}}{=} \emptyset
\]

\[
\gamma_{\text{Int}}([a, b]) \overset{\text{def}}{=} \{ z \in \mathbb{Z} \mid a \leq z \text{ and } z \leq b \}
\]
All the other operators are strict: they return \( \bot \) if one of their arguments is \( \bot \).

\[ + \# ([a, b], [c, d]) = [a + c, b + d] \]
\[ - \# ([a, b], [c, d]) = [a - d, b - c] \]
\[ \times \# ([a, b], [c, d]) = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \]

\[ [+]_{\text{op}} ([a, b], [c, d], [e, f]) = \rho(\max(c, a - f), \min(d, b - e)), \]
\[ ... \]
\[ \leq \#_{\text{comp}} ([a, b], [c, d]) = ([a, b] \cap_{\text{Int}} [-\infty, d - 1], [a + 1, +\infty] \cap_{\text{Int}} [c, d]) \]
\[ \neq \#_{\text{comp}} ([a, b], [c, d]) = ? \text{exercise...} \]
\[ \text{const}(n)\# = [n, n] \]
Convergence problem

Such a lattice does not satisfy the ascending chain condition.

Example of infinite increasing chain:
\[ \perp \sqsubseteq [0, 0] \sqsubseteq [0, 1] \sqsubseteq \cdots \sqsubseteq [0, n] \sqsubseteq \cdots \]

Solution: dynamic approximation
Lemma

Let \((A, \sqsubseteq, \sqcup, \sqcap)\) a complete lattice and \(f\) a monotone operator on \(A\). If \(a\) is a post-fixpoint of \(f\) (i.e. \(f(a) \sqsubseteq a\)), then \(\text{lfp}(f) \sqsubseteq a\).

We may want to compute an over-approximation of \(\text{lfp}(f)\) in the following cases:

- The lattice does not satisfy the ascending chain condition, the iteration \(\bot, f(\bot), \ldots, f^n(\bot), \ldots\) may never terminates.
- The ascending chain condition is satisfied but the iteration chain is too long to allow an efficient computation.
- Finally, some abstractions do not verify the “good approximation space” criterion previously described. The underlying lattice is not complete, the limits of the ascending iterations do not necessarily belong to the abstraction domain.
Widening

Idea: the standard iteration is of the form

\[ x^0 = \bot, x^{n+1} = F(x^n) = x^n \sqcup F(x^n) \]

We will replace it by something of the form

\[ y^0 = \bot, y^{n+1} = y^n \nabla F(y^n) \]

such that

(i) \( (y^n) \) is increasing,
(ii) \( x^n \sqsubseteq y^n \), for all \( n \),
(iii) and \( (y^n) \) stabilizes after a finite number of steps.

But we also want a \( \nabla \) operator that is independent of \( F \).
Widening: definition

A widening is an operator $\nabla : L \times L \to L$ such that

1. $\forall x, x' \in L, x \sqcup x' \sqsubseteq x \nabla x'$ (implies (i) & (ii))
2. If $x^0 \sqsubseteq x^1 \sqsubseteq \ldots$ is an increasing chain, then the increasing chain $y^0 = x^0, y^{n+1} = y^n \nabla x^{n+1}$ stabilizes after a finite number of steps (implies (iii)).

Usage: we replace $x^0 = \perp, x^{n+1} = F(x^n)$ by $y^0 = \perp, y^{n+1} = y^n \nabla F(y^n)$
Widening: theorem

Theorem

Let

- $L$ be a complete lattice,
- $F : L \rightarrow L$ be a monotone function and
- $\nabla : L \times L \rightarrow L$ a widening operator.

Then, the chain $y^0 = \bot, y^{n+1} = y^n \nabla F(y^n)$ stabilizes after a finite number of steps at a post-fixpoint $y$ of $F$.

Corollary: $\text{lfp}(F) \sqsubseteq y$. 
Scheme

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

decreasing iteration with \( \Delta \)

increasing iteration with \( \nabla \)
Example: widening on intervals

Idea: as soon as a bound is not stable, we extrapolate it by $+\infty$ (or $-\infty$). After such an extrapolation, the bound can’t move any more.

Definition:

\[
[a, b] \nabla_{\text{Int}} [a', b'] = [\begin{cases} \infty & \text{if } a' < a \text{ then } \infty \text{ else } a, \\ a & \text{if } b' > b \text{ then } +\infty \text{ else } b \end{cases}]
\]

\[
\bot \nabla_{\text{Int}} [a', b'] = [a', b']
\]

\[
I \nabla_{\text{Int}} \bot = I
\]

Examples:

\[
[-3, 4] \nabla_{\text{Int}} [-3, 2] = [-3, 4]
\]

\[
[-3, 4] \nabla_{\text{Int}} [-3, 5] = [-3, +\infty]
\]
Example

\begin{align*}
  &x := 100; \\
  &\text{while } 0 < x \{ \\
  &\quad x := x - 1; \\
  &\}
\end{align*}

\begin{align*}
  X_1 &= [100, 100] \sqcup_{\text{Int}} (X_2 - \# [1, 1]) \\
  X_2 &= [1, +\infty] \cap_{\text{Int}} X_1 \\
  X_3 &= [-\infty, 0] \cap_{\text{Int}} X_1
\end{align*}
Example: without widening

\[ X_1 = [100, 100] \uplus_{\text{Int}} (X_2 - \# [1, 1]) \]
\[ X_2 = [1, +\infty] \cap_{\text{Int}} X_1 \]
\[ X_3 = [-\infty, 0] \cap_{\text{Int}} X_1 \]

Iteration strategy: \( 1 \to 2 \to 3 \to 1 \to 2 \to \cdots \)

\[ X_1^0 = \bot \quad X_1^{n+1} = [100, 100] \uplus_{\text{Int}} (X_2^n - \# [1, 1]) \]
\[ X_2^0 = \bot \quad X_2^{n+1} = [1, +\infty] \cap_{\text{Int}} X_1^{n+1} \]
\[ X_3^0 = \bot \quad X_3^{n+1} = [-\infty, 0] \cap_{\text{Int}} X_1^{n+1} \]
Example: without widening

\[ X_1 = [100, 100] \sqcup_{\text{Int}} (X_2 \# [1, 1]) \]
\[ X_2 = [1, +\infty] \cap_{\text{Int}} X_1 \]
\[ X_3 = [-\infty, 0] \cap_{\text{Int}} X_1 \]

Iteration strategy: 1 → 2 → 3 → 1 → 2 → ···

\[ X_1^0 = \bot \quad X_1^{n+1} = [100, 100] \sqcup_{\text{Int}} (X_2^n \# [1, 1]) \]
\[ X_2^0 = \bot \quad X_2^{n+1} = [1, +\infty] \cap_{\text{Int}} X_1^{n+1} \]
\[ X_3^0 = \bot \quad X_3^{n+1} = [-\infty, 0] \cap_{\text{Int}} X_1^{n+1} \]
Example: with widening at each nodes of the cfg

\[ X_1 = [100, 100] \sqcup_{\text{Int}} (X_2 - \# [1, 1]) \]
\[ X_2 = [1, +\infty] \cap_{\text{Int}} X_1 \]
\[ X_3 = [-\infty, 0] \cap_{\text{Int}} X_1 \]

Iteration strategy: \( 1 \to 2 \to 3 \to 1 \to 2 \to \cdots \)

\[ X^0_1 = \perp \]
\[ X^{n+1}_1 = X^n_1 \nabla_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X^n_2 - \# [1, 1])) \]
\[ X^0_2 = \perp \]
\[ X^{n+1}_2 = X^n_2 \nabla_{\text{Int}} ([1, +\infty] \cap_{\text{Int}} X^{n+1}_1) \]
\[ X^0_3 = \perp \]
\[ X^{n+1}_3 = X^n_3 \nabla_{\text{Int}} ([-\infty, 0] \cap_{\text{Int}} X^{n+1}_1) \]

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( \perp )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_2 )</td>
<td>( \perp )</td>
</tr>
<tr>
<td>( X_3 )</td>
<td>( \perp )</td>
</tr>
</tbody>
</table>
Example: with widening at each nodes of the CFG

\[
X_1 = [100, 100] \sqcup_{\text{Int}} (X_2 - \# [1, 1])
\]
\[
X_2 = [1, +\infty] \cap_{\text{Int}} X_1
\]
\[
X_3 = [-\infty, 0] \cap_{\text{Int}} X_1
\]

Iteration strategy: 1 → 2 → 3 → 1 → 2 → ···

\[
X_0^1 = \perp \quad X_1^{n+1} = X_1^n \nabla_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X_2^n - \# [1, 1]))
\]
\[
X_0^2 = \perp \quad X_2^{n+1} = X_2^n \nabla_{\text{Int}} ([1, +\infty] \cap_{\text{Int}} X_1^{n+1})
\]
\[
X_0^3 = \perp \quad X_3^{n+1} = X_3^n \nabla_{\text{Int}} ([-\infty, 0] \cap_{\text{Int}} X_1^{n+1})
\]

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$[100, 100]$</th>
<th>$[-\infty, 100]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_2$</td>
<td>$\perp$</td>
<td>$[100, 100]$</td>
<td>$[-\infty, 100]$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td>$[-\infty, 0]$</td>
</tr>
</tbody>
</table>
Improving fixpoint approximation

Idea: iterating a little more may help...

**Theorem**

Let \((A, \sqsubseteq, \sqcup, \sqcap)\) a complete lattice, \(f\) a monotone operator on \(A\) and \(a\) a post-fixpoint of \(f\). The chain \((x_n)_n\) defined by
\[
\begin{align*}
x_0 &= a \\
x_{k+1} &= f(x_k)
\end{align*}
\]
admits for limit \((\sqcap \{x_n\})\) the greatest fixpoint of \(f\) lower than \(a\) (written \(\text{gfp}_a(f)\)). In particular, \(\text{lfp}(f) \sqsubseteq \sqcap \{x_n\}\). Each intermediate step is a correct approximation:
\[
\forall k, \text{lfp}(f) \sqsubseteq \text{gfp}_a(f) \sqsubseteq x_k \sqsubseteq a
\]
Narrowing: definition

A narrowing is an operator $\Delta : L \times L \to L$ such that

- $\forall x, x' \in L, x' \sqsubseteq x \Delta x' \sqsubseteq x$
- If $x^0 \sqsupseteq x^1 \sqsupseteq \ldots$ is a decreasing chain, then the chain $y^0 = x^0, y^{n+1} = y^n \Delta x^{n+1}$ stabilizes after a finite number of steps.
Theorem

If $\Delta$ is a narrowing operator on a poset $(A, \sqsubseteq)$, if $f$ is a monotone operator on $A$ and $a$ is a post-fixpoint of $f$ then the chain $(x_n)_n$ defined by

\[
\begin{align*}
    x_0 &= a \\
    x_{k+1} &= x_k \Delta f(x_k)
\end{align*}
\]

stabilizes after a finite number of steps on a post-fixpoint of $f$ lower than $a$. 

Narrowing: decreasing iteration
Narrowing on intervals

\[
[a, b] \Delta_{\text{Int}} [c, d] = [\text{if } a = -\infty \text{ then } c \text{ else } a ; \text{ if } b = +\infty \text{ then } d \text{ else } b]
\]

\[
I \Delta_{\text{Int}} \bot = \bot
\]

\[
\bot \Delta_{\text{Int}} I = \bot
\]

Intuition: we only improve infinite bounds.

In practice: a few standard iterations already improve a lot the result that has been obtained after widening...

- Assignments by constants and conditional guards make the decreasing iterations efficient: they filter the (too big) approximations computed by the widening
Example: with narrowing at each node of the cfg

\[X_1 = [100, 100] \uplus_{\text{Int}} (X_2 - \# [1, 1])\]
\[X_2 = [1, +\infty] \cap_{\text{Int}} X_1\]
\[X_3 = [-\infty, 0] \cap_{\text{Int}} X_1\]

Iteration strategy: \(1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow \cdots\)

\[X_1^0 = [-\infty, 100] \quad X_1^{n+1} = X_1^n \Delta_{\text{Int}} \left([100, 100] \uplus_{\text{Int}} (X_2^n - \# [1, 1])\right)\]
\[X_2^0 = [-\infty, 100] \quad X_2^{n+1} = X_2^n \Delta_{\text{Int}} \left([1, +\infty] \cap_{\text{Int}} X_1^{n+1}\right)\]
\[X_3^0 = [-\infty, 0] \quad X_3^{n+1} = X_3^n \Delta_{\text{Int}} \left([-\infty, 0] \cap_{\text{Int}} X_1^{n+1}\right)\]

<table>
<thead>
<tr>
<th>(X_i)</th>
<th>([-\infty, 100])</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_1)</td>
<td>([-\infty, 100])</td>
</tr>
<tr>
<td>(X_2)</td>
<td>([-\infty, 100])</td>
</tr>
<tr>
<td>(X_3)</td>
<td>([-\infty, 0])</td>
</tr>
</tbody>
</table>
Example: with narrowing at each node of the cfg

\[ X_1 = [100, 100] \cup_{\text{Int}} (X_2 - \# [1, 1]) \]
\[ X_2 = [1, +\infty] \cap_{\text{Int}} X_1 \]
\[ X_3 = [-\infty, 0] \cap_{\text{Int}} X_1 \]

Iteration strategy: 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow \cdots

\[ X_1^0 = [-\infty, 100] \quad X_1^{n+1} = X_1^n \Delta_{\text{Int}} ([100, 100] \cup_{\text{Int}} (X_2^n - \# [1, 1])) \]
\[ X_2^0 = [-\infty, 100] \quad X_2^{n+1} = X_2^n \Delta_{\text{Int}} ([1, +\infty] \cap_{\text{Int}} X_1^{n+1}) \]
\[ X_3^0 = [-\infty, 0] \quad X_3^{n+1} = X_3^n \Delta_{\text{Int}} ([-\infty, 0] \cap_{\text{Int}} X_1^{n+1}) \]

<table>
<thead>
<tr>
<th></th>
<th>([-\infty, 100])</th>
<th>([-\infty, 100])</th>
<th>([0, 100])</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_1)</td>
<td>([-\infty, 100])</td>
<td>([-\infty, 100])</td>
<td>([0, 100])</td>
</tr>
<tr>
<td>(X_2)</td>
<td>([-\infty, 100])</td>
<td>([1, 100])</td>
<td>([1, 100])</td>
</tr>
<tr>
<td>(X_3)</td>
<td>([-\infty, 0])</td>
<td>([-\infty, 0])</td>
<td>([0, 0])</td>
</tr>
</tbody>
</table>
The particular case of an equation system

Consider a system

\[
\begin{align*}
x_1 &= f_1(x_1, \ldots, x_n) \\
\vdots \\
x_n &= f_n(x_1, \ldots, x_n)
\end{align*}
\]

with \(f_1, \ldots, f_n\) monotones.

Standard iteration:

\[
\begin{align*}
x_{i+1}^1 &= f_1(x_i^1, \ldots, x_i^n) \\
x_{i+1}^2 &= f_2(x_i^1, \ldots, x_i^n) \\
&\quad \vdots \\
x_{i+1}^n &= f_n(x_i^1, \ldots, x_i^n)
\end{align*}
\]

Standard iteration with widening:

\[
\begin{align*}
x_{i+1}^1 &= x_i^1 \triangledown f_1(x_i^1, \ldots, x_i^n) \\
x_{i+1}^2 &= x_i^2 \triangledown f_2(x_i^1, \ldots, x_i^n) \\
&\quad \vdots \\
x_{i+1}^n &= x_i^n \triangledown f_n(x_i^1, \ldots, x_i^n)
\end{align*}
\]
The particular case of an equation system

\[ \begin{align*}
  x_1 &= f_1(x_1, \ldots, x_n) \\
  \vdots \\
  x_n &= f_n(x_1, \ldots, x_n)
\end{align*} \]

It is sufficient (and generally more precise) to use \( \nabla \) for a selection of index \( W \) such that each dependence cycle in the system goes through at least one point in \( W \).

\[ \forall k = 1..n, \quad x_k^{i+1} = x_k^i \nabla f_k(x_1^i, \ldots, x_n^i) \quad \text{if } k \in W \\
  f_k(x_1^i, \ldots, x_n^i) \quad \text{otherwise} \]

**Chaotic iteration:** at each step, we use only one equation, without forgetting one for ever.

**Delayed widening:** It is generally better to wait a few standard iterations before launching the widenings.
Polyhedral abstract interpretation

*Automatic discovery of linear restraints among variables of a program.*
P. Cousot and N. Halbwachs. POPL’78.

Polyhedral analysis seeks to discover invariant linear equality and inequality relationships among the variables of an imperative program.
Convex polyhedra

A convex polyhedron can be defined algebraically as the set of solutions of a system of linear inequalities.

Geometrically, it can be defined as a finite intersection of half-spaces.
Polyhedral analysis

State properties are over-approximated by convex polyhedra in \( \mathbb{Q}^2 \).

\[
x = 0; \quad y = 0;
\]

\[
\textbf{while} \ (x<6) \{
    \textbf{if} \ (?) \{
        y = y+2;
    \};
    x = x+1;
\}
\]
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

$$x = 0; y = 0;$$

$$\{x = 0 \land y = 0\}$$

$$\textbf{while } (x<6) \{$$

$$\textbf{if } (?) \{$$

$$\{x = 0 \land y = 0\}$$

$$y = y+2;$$

};

$$x = x+1;$$

}
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
x = 0; \ y = 0;
\{x = 0 \land y = 0\}
\]

\[
\text{while} \ (x<6) \ {}\{
\text{if} \ (?) \ {}\{
\{x = 0 \land y = 0\}
\ y = y+2;\n\{x = 0 \land y = 2\}
\};
\{x = 0 \land y = 0\} \uplus \{x = 0 \land y = 2\}
\}
\]

At junction points, we over-approximates union by a convex union.

\[
x = x+1;
\]

\}
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
x = 0; \ y = 0;
{x = 0 \land y = 0}
\]

\[
\textbf{while} \ (x<6) \ {\}
\]
\[
\textbf{if} \ (?) \ {\}
\]
\[
{x = 0 \land y = 0}
\]
\[
y = y+2;
{x = 0 \land y = 2}
\]
\[
}; \ {\}
\]
\[
{x = 0 \land 0 \leq y \leq 2}
\]
\[
\textbf{while} \ (x<6) \ {\}
\]
\[
{x = x+1;}
\]

At junction points, we over-approximates union by a convex union.
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
x = 0; \quad y = 0;
\{ x = 0 \land y = 0 \}
\]

\[
\text{while (x<6) } \{
\quad \text{if (?) } \{
\quad \quad \{ x = 0 \land y = 0 \}
\quad \quad y = y+2;
\quad \quad \{ x = 0 \land y = 2 \}
\quad \};
\quad \{ x = 0 \land 0 \leq y \leq 2 \}
\]

\[
x = x+1;
\quad \{ x = 1 \land 0 \leq y \leq 2 \}
\]

}
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
x = 0; \quad y = 0;
\]

\[
\{x = 0 \land y = 0\} \cup \{x = 1 \land 0 \leq y \leq 2\}
\]

\[
\textbf{while} \ (x < 6) \ {\{}
\]

\[
\textbf{if} \ (\?) \ {\{}
\]

\[
x = 0 \land y = 0
\]

\[
y = y + 2;
\]

\[
\{x = 0 \land y = 2\}
\]

\[
\}
\]

\[
\textbf{if} \ (\?) \ {\{}
\]

\[
x = x + 1;
\]

\[
\{x = 1 \land 0 \leq y \leq 2\}
\]

\[
\}
\]
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $Q^2$.

$$x = 0; \ y = 0;$$
$$\{x \leq 1 \land 0 \leq y \leq 2x\}$$

$$\textbf{while} \ (x<6) \ {\{ \textbf{if} \ (?) \ {\{ x = 0 \land y = 0 \} \ y = y+2; \ {\{ x = 0 \land y = 2 \} } \}; \ {\{ x = 0 \land 0 \leq y \leq 2 \} } x = x+1; \ {\{ x = 1 \land 0 \leq y \leq 2 \} } \}$$
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
x = 0; \quad y = 0; \\
\{x \leq 1 \land 0 \leq y \leq 2x\}
\]

\[
\text{while} \ (x < 6) \{
\quad \text{if} \ (? \}) \{
\quad \{x \leq 1 \land 0 \leq y \leq 2x\}
\quad y = y + 2; \\
\quad \{x = 0 \land y = 2\}
\}; \\
\quad \{x = 0 \land 0 \leq y \leq 2\}
\]

\[
x = x + 1; \\
\{x = 1 \land 0 \leq y \leq 2\}
\]
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
\begin{align*}
  x &= 0; \ y = 0; \\
  &\{x \leq 1 \land 0 \leq y \leq 2x\} \\
\end{align*}
\]

\[
\begin{align*}
  \textbf{while} \ (x<6) \ {\{} \\
  &\textbf{if} \ (?) \ {\{} \\
  &\quad \{x \leq 1 \land 0 \leq y \leq 2x\} \\
  &\quad y = y+2; \\
  &\quad \{x \leq 1 \land 2 \leq y \leq 2x + 2\} \\
  &\}; \\
  &\{x = 0 \land 0 \leq y \leq 2\} \\
  &\{x = x+1; \\
  &\quad \{x = 1 \land 0 \leq y \leq 2\} \\
  &\}
\end{align*}
\]
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
x = 0; \quad y = 0;
\{
x \leq 1 \land 0 \leq y \leq 2x
\}
\]

\[
\text{while } (x<6) \{
\quad \text{if } (?) \{
\quad \quad \quad \{x \leq 1 \land 0 \leq y \leq 2x\}
\quad \quad y = y+2;
\quad \quad \quad \{x \leq 1 \land 2 \leq y \leq 2x + 2\}
\quad \}:
\quad \{x \leq 1 \land 0 \leq y \leq 2x\}
\quad \cup\{x \leq 1 \land 2 \leq y \leq 2x + 2\}
\quad x = x+1;
\quad \{x = 1 \land 0 \leq y \leq 2\}
\}
\]
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $Q^2$.

\[
x = 0; \ y = 0;
\{x \leq 1 \land 0 \leq y \leq 2x\}
\]

\[
\textbf{while } (x<6) \{ \\
\quad \textbf{if } (?) \{ \\
\quad \quad \{x \leq 1 \land 0 \leq y \leq 2x\}
\quad \}\; y = y+2;
\quad \{x \leq 1 \land 2 \leq y \leq 2x + 2\}
\}
\{0 \leq x \leq 1 \land 0 \leq y \leq 2x + 2\}
\]

\[
x = x+1;
\{x = 1 \land 0 \leq y \leq 2\}
\}
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $Q^2$.

\[
x = 0; \quad y = 0;
\]
\[
\{x \leq 1 \land 0 \leq y \leq 2x\}
\]

\[
\textbf{while} \ (x<6) \ {\}{
\quad \textbf{if} \ (?) \ {\}{
\quad \quad \{x \leq 1 \land 0 \leq y \leq 2x\}
\quad \quad y = y+2;
\quad \quad \{x \leq 1 \land 2 \leq y \leq 2x + 2\}
\quad \};
\quad \{0 \leq x \leq 1 \land 0 \leq y \leq 2x + 2\}
\]

\[
x = x+1;
\]
\[
\{1 \leq x \leq 2 \land 0 \leq y \leq 2x\}
\]
\]
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $Q^2$.

$$x = 0; \ y = 0;$$
$$\{x \leq 1 \ \land \ 0 \leq y \leq 2x\}$$
$$\lor \{x \leq 2 \ \land \ 0 \leq y \leq 2x\}$$

```cpp
while (x<6) {
    if (?) {
        \{x \leq 1 \ \land \ 0 \leq y \leq 2x\}
        y = y+2;
        \{x \leq 1 \ \land \ 2 \leq y \leq 2x + 2\}
    }
    \{0 \leq x \leq 1 \ \land \ 0 \leq y \leq 2x + 2\}
    x = x+1;
    \{1 \leq x \leq 2 \ \land \ 0 \leq y \leq 2x\}
}
```

At loop headers, we use heuristics (widening) to ensure finite convergence.
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $Q^2$.

\[
x = 0; \quad y = 0; \quad \{0 \leq y \leq 2x\}
\]

\[
\text{while } (x<6) \{
\quad \text{if (?) } \{
\quad \quad \{x \leq 1 \land 0 \leq y \leq 2x\}
\quad \quad y = y+2;
\quad \quad \{x \leq 1 \land 2 \leq y \leq 2x + 2\}
\quad \};
\quad \{0 \leq x \leq 1 \land 0 \leq y \leq 2x + 2\}
\]

\[
x = x+1;
\quad \{1 \leq x \leq 2 \land 0 \leq y \leq 2x\}
\]

\}

At loop headers, we use heuristics (widening) to ensure finite convergence.
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $Q^2$.

- $x = 0; \ y = 0;$
  - $\{0 \leq y \leq 2x\}$

By propagation we obtain a post-fixpoint

```plaintext
while (x<6) {
    if (?) {
        $\{0 \leq y \leq 2x \land x \leq 5\}$
        $y = y+2;$
        $\{2 \leq y \leq 2x + 2 \land x \leq 5\}$
    }
    $\{0 \leq y \leq 2x + 2 \land 0 \leq x \leq 5\}$
}
$x = x+1;$
$\{0 \leq y \leq 2x \land 1 \leq x \leq 6\}$
}```
Polyhedral analysis

State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.

\[
x = 0; \ y = 0;
\{0 \leq y \leq 2x \land x \leq 6\}
\]

\begin{verbatim}
while (x<6) {
  if (?) {
    {0 \leq y \leq 2x \land x \leq 5}
    y = y+2;
    {2 \leq y \leq 2x + 2 \land x \leq 5}
  }
  {0 \leq y \leq 2x + 2 \land 0 \leq x \leq 5}
  x = x+1;
  {0 \leq y \leq 2x \land 1 \leq x \leq 6}
}
{0 \leq y \leq 2x \land 6 = x}
\end{verbatim}

By propagation we obtain a post-fixpoint which is enhanced by downward iteration.
Polyhedral analysis

A more complex example.

\[
x = 0; \ y = A;
\{ A \leq y \leq 2x + A \land x \leq N \}
\]

\[
\textbf{while} \ (x<N) \ {\}
\textbf{if} \ (?) \ {\}
\quad \{ A \leq y \leq 2x + A \land x \leq N - 1 \}
\quad y = y+2;
\quad \{ A + 2 \leq y \leq 2x + A + 2 \land x \leq N - 1 \}
\}
\quad \{ A \leq y \leq 2x + A + 2 \land 0 \leq x \leq N - 1 \}
\}
\textbf{while} \ (x<N) \ {\}
\quad x = x+1;
\quad \{ A \leq y \leq 2x + A \land 1 \leq x \leq N \}
\}
\quad \{ A \leq y \leq 2x + A \land N = x \}
\]
The four polyhedra operations

- $\ominus \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$: convex union
  - over-approximates the concrete union at junction points

- $\cap \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$: intersection
  - over-approximates the concrete intersection after a conditional instruction

- $\llbracket x := e \rrbracket \in \mathbb{P}_n \rightarrow \mathbb{P}_n$: affine transformation
  - over-approximates the assignment of a variable by a linear expression

- $\nabla \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$: widening
  - ensures (and accelerates) convergence of (post-)fixpoint iteration
  - includes heuristics to infer loop invariants

---

```plaintext
x = 0; y = 0;
P₀ = \llbracket y := 0 \rrbracket \llbracket x := 0 \rrbracket (Q^2) \nabla P₄

while (x<6) {
  if (?) {
    P₁ = P₀ \cap \{x < 6\}
    y = y+2;
P₂ = \llbracket y := y + 2 \rrbracket (P₁)
  }
  P₃ = P₁ \uplus P₂
  x = x+1;
P₄ = \llbracket x := x + 1 \rrbracket (P₃)
}
P₅ = P₀ \cap \{x \geq 6\}
```
Library for manipulating polyhedra

- Parma Polyhedra Library\(^1\) (PPL), NewPolka: complex C/C++ libraries
- They rely on the Double Description Method
  - Polyhedra are managed using two representations in parallel
    - By set of inequalities
      \[
P = \left\{ (x, y) \in \mathbb{Q}^2 \right\} \begin{align*}
        x & \geq -1 \\
        x - y & \geq -3 \\
        2x + y & \geq -2 \\
        x + 2y & \geq -4
      \end{align*}
    
    - By set of generators
      \[
P = \left\{ \lambda_1 s_1 + \lambda_2 s_2 + \lambda_3 s_3 + \mu_1 r_1 + \mu_2 r_2 \in \mathbb{Q}^2 \right\} \begin{align*}
        \lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2 & \in \mathbb{R}^+ \\
        \lambda_1 + \lambda_2 + \lambda_3 & = 1
      \end{align*}
    
- Operations efficiency strongly depends on the chosen representations, so they keep both

\(^1\)Previous tutorial on polyhedra partially comes from http://www.cs.unipr.it/ppl/
References (1)

A few articles

- a short formal introduction
  
  \textit{P. Cousot and R. Cousot. Basic Concepts of Abstract Interpretation.}  
  \url{http://www.di.ens.fr/~cousot/COUSOTpapers/WCC04.shtml}

- technical but very complete (the logic programming part is optional):
  
  \textit{P. Cousot and R. Cousot. Abstract Interpretation and Application to Logic Programs.}  
  \url{http://www.di.ens.fr/~cousot/COUSOTpapers/JLP92.shtml}

- a nice application of abstract interpretation theory to verify airbus flight commands
  
  \textit{P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE Analyser.}  
  \url{http://www.di.ens.fr/~cousot/COUSOTpapers/ESOP05.shtml}
References (2)

On the web:

- informal presentation of AI with nice pictures
  \[\text{http://www.di.ens.fr/}~\text{cousot/AI/IntroAbsInt.html}\]

- a short abstract of various works around AI
  \[\text{http://www.di.ens.fr/}~\text{cousot/AI/}\]

- very complete lecture notes
  \[\text{http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/}\]