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 $(\mathtt{x}, \mathtt{y})$ 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;
      {                    }
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 $(\mathtt{x}, \mathtt{y})$ 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 (?) {
      {                     }
    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 $(\mathtt{x},\mathtt{y})$ 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;
      {               }
  };
      {                          }
  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 $(\mathtt{x},\mathtt{y})$ 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;
      {(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 $(\mathtt{x},\mathtt{y})$ 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;
      {(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 $(\mathtt{x}, \mathtt{y})$ 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;
      {(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 $(\mathtt{x},\mathtt{y})$ 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), (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)                 }
}
```

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 $(\mathtt{x},\mathtt{y})$ 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),(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 $(\mathtt{x}, \mathtt{y})$ 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),(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 $(\mathtt{x},\mathtt{y})$ 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),(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 $(\mathtt{x}, \mathtt{y})$ 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),(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 $(\mathtt{x},\mathtt{y})$ 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), (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

$$
\begin{array}{rcl}
P & ::= & x \, C \, 0 \wedge y \, C \, 0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

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

$\mathtt{x} = 0 \wedge \mathtt{y} = 0$

```
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.

Approximation

- The set of manipulated properties may be restricted to ensure computability of the semantics.
  Example : sign of variables

$$
\begin{array}{lcl}
P & ::= & x\,C\,0 \wedge y\,C\,0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x = 0 ∧ y = 0
while (x<6) {
  if (?) {
        x = 0 ∧ 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

$$\begin{array}{rcl} P & ::= & x\,C\,0 \wedge y\,C\,0 \\ C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant \end{array}$$

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

```
x = 0; y = 0;
        x = 0 ∧ y = 0
while (x<6) {
  if (?) {
        x = 0 ∧ y = 0
    y = y+2;
        x = 0 ∧ y > 0   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

$$
\begin{array}{rcl}
P & ::= & x\, C\, 0 \wedge y\, C\, 0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x = 0 ∧ y = 0
while (x<6) {
  if (?) {
        x = 0 ∧ y = 0
    y = y+2;
        x = 0 ∧ y > 0
  };
        x = 0 ∧ y ⩾ 0
  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

$$
\begin{array}{rcl}
P & ::= & x\,C\,0 \wedge y\,C\,0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x = 0 ∧ y = 0
while (x<6) {
  if (?) {
        x = 0 ∧ y = 0
    y = y+2;
        x = 0 ∧ y > 0
  };
        x = 0 ∧ y ⩾ 0
  x = x+1;
        x > 0 ∧ y ⩾ 0  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

$$\begin{array}{rcl} P & ::= & x\,C\,0 \wedge y\,C\,0 \\ C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant \end{array}$$

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

```
x = 0; y = 0;
      x ⩾ 0 ∧ y ⩾ 0
while (x<6) {
  if (?) {
      x = 0 ∧ y = 0
    y = y+2;
      x = 0 ∧ y > 0
  };
      x = 0 ∧ y ⩾ 0
  x = x+1;
      x > 0 ∧ y ⩾ 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

$$
\begin{array}{rcl}
P & ::= & x\,C\,0 \wedge y\,C\,0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x ⩾ 0 ∧ y ⩾ 0
while (x<6) {
  if (?) {
        x ⩾ 0 ∧ y ⩾ 0
    y = y+2;
        x = 0 ∧ y > 0
  };
        x = 0 ∧ y ⩾ 0
  x = x+1;
        x > 0 ∧ y ⩾ 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

$$
\begin{array}{rcl}
P & ::= & x\, C\, 0 \wedge y\, C\, 0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x ⩾ 0 ∧ y ⩾ 0
while (x<6) {
  if (?) {
        x ⩾ 0 ∧ y ⩾ 0
    y = y+2;
        x ⩾ 0 ∧ y > 0
  };
        x = 0 ∧ y ⩾ 0
  x = x+1;
        x > 0 ∧ y ⩾ 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

$$
\begin{array}{rcl}
P & ::= & x\,C\,0 \wedge y\,C\,0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x ⩾ 0 ∧ y ⩾ 0
while (x<6) {
  if (?) {
        x ⩾ 0 ∧ y ⩾ 0
    y = y+2;
        x ⩾ 0 ∧ y > 0
  };
        x ⩾ 0 ∧ y ⩾ 0
  x = x+1;
        x > 0 ∧ y ⩾ 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

$$
\begin{array}{lcl}
P & ::= & x\, C\, 0 \wedge y\, C\, 0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x ⩾ 0 ∧ y ⩾ 0
while (x<6) {
  if (?) {
        x ⩾ 0 ∧ y ⩾ 0
    y = y+2;
        x ⩾ 0 ∧ y > 0
  };
        x ⩾ 0 ∧ y ⩾ 0
  x = x+1;
        x > 0 ∧ y ⩾ 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

$$
\begin{array}{rcl}
P & ::= & x\, C\, 0 \wedge y\, C\, 0 \\
C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant
\end{array}
$$

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

```
x = 0; y = 0;
        x ⩾ 0 ∧ y ⩾ 0
while (x<6) {
  if (?) {
        x ⩾ 0 ∧ y ⩾ 0
    y = y+2;
        x ⩾ 0 ∧ y > 0
  };
        x ⩾ 0 ∧ y ⩾ 0
  x = x+1;
        x > 0 ∧ y ⩾ 0
}
        x ⩾ 0 ∧ y ⩾ 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

```
assert(T.length=100); i=1;
```
$$\{i \in [1,100]\}$$
```
while (i<T.length) {
```
$$\{i \in [1,99]\}$$
```
    p = T[i]; j = i-1;
```
$$\{i \in [1,99],\ j \in [-1,98]\}$$
```
    while (0<=j and T[j]>p) {
```
$$\{i \in [1,99],\ j \in [0,98]\}$$
```
        T[j]=T[j+1]; j = j-1;
```
$$\{i \in [1,99],\ j \in [-1,97]\}$$
```
    };
```
$$\{i \in [1,99],\ j \in [-1,98]\}$$
```
    T[j+1]=p; i = i+1;
```
$$\{i \in [2,100],\ j = [-1,98]\}$$
```
};
```

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

```
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;

};
```

$\{1 \leqslant i \leqslant T.length\}$

$\{1 \leqslant i \leqslant T.length - 1\}$

$\{1 \leqslant i \leqslant T.length - 1 \ \wedge\ -1 \leqslant j \leqslant i - 1\}$

$\{1 \leqslant i \leqslant T.length - 1 \ \wedge\ 0 \leqslant j \leqslant i - 1\}$

$\{1 \leqslant i \leqslant T.length - 1 \ \wedge\ -1 \leqslant j \leqslant i - 2\}$

$\{1 \leqslant i \leqslant T.length - 1 \ \wedge\ -1 \leqslant j \leqslant i - 1\}$

$\{2 \leqslant i \leqslant T.length + 1 \ \wedge\ -1 \leqslant j \leqslant i - 2\}$

Collecting Semantics

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

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

Theorem

$[\![p]\!]^{col}$ *may be characterized as the least fixpoint of the following equation system.*

$$\forall k \in labels(p),\ X_k = X^{init}_k \cup \bigcup_{(k',i,k)\in p} [\![i]\!]\,(X_{k'})$$

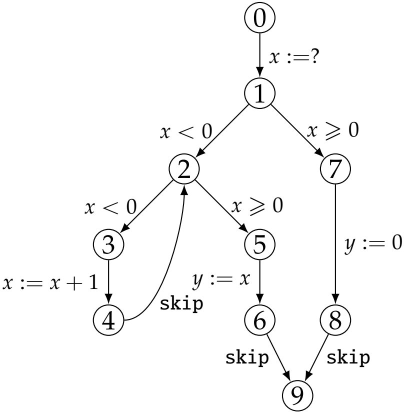
with $X^{\text{init}}_k = \begin{cases} Env & \text{if } k = k_{\text{init}} \\ \emptyset & \text{otherwise} \end{cases}$

and

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

Example

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



$$
\begin{aligned}
X_0 &= Env \\
X_1 &= [\![x :=?]\!]\,(X_0) \\
X_2 &= [\![x < 0]\!]\,(X_1) \cup X_4 \\
X_3 &= [\![x < 0]\!]\,(X_2) \\
X_4 &= [\![x := x+1]\!]\,(X_3) \\
X_5 &= [\![x \geqslant 0]\!]\,(X_2) \\
X_6 &= [\![y := x]\!]\,(X_5) \\
X_7 &= [\![x \geqslant 0]\!]\,(X_1) \\
X_8 &= [\![y := 0]\!]\,(X_7) \\
X_9 &= X_6 \cup X_8
\end{aligned}
$$

Collecting semantics and exact analysis

The $(X_k)_{i=1..N}$ are hence specified as the least solution of a fixpoint equation system

$$X_k = F_k(X_1, X_2, \ldots, X_N) \ \ , \ \ k \in labels(p)$$

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

$$X_k^0 = \emptyset \ \ , \ \ X_k^{n+1} = F_k(X_1^n, X_2^n, \ldots, X_N^n)$$

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 $(\mathcal{P}(Env)^N, \subseteq, \cup, \cap)$
or limit of $X^0 = \bot, X^{n+1} = F(X^n)$

Approximate analysis:

- Static approximation: we replace the concrete lattice $(\mathcal{P}(Env), \subseteq, \cup, \cap)$ by an abstract lattice $(L^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp)$
  - whose elements can be (efficiently) represented in computers,
  - in which we know how to compute $\sqcup^\sharp, \sqcap^\sharp, \sqsubseteq^\sharp, \ldots$

  and we "transpose" the equation $X = F(X)$ of $\mathcal{P}(Env)^N$ into $(L^\sharp)^N$.
- Dynamic approximation: when $L^\sharp$ does not verifies 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).

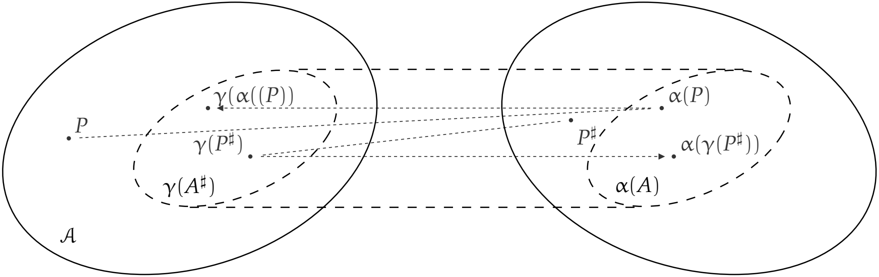
What is a *good* approximation space?

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* $\alpha \in L_1 \to L_2$ *and* $\gamma \in L_2 \to L_1$ *is a Galois connection if it verifies the condition*

$$\forall x_1 \in L_1, \forall x_2 \in L_2,\ \alpha(x_1) \sqsubseteq_2 x_2 \iff x_1 \sqsubseteq_1 \gamma(x_2)$$

A complete lattice $(A^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp)$ is *a good approximation space* if there exists a Galois connection between $(\mathcal{A}, \subseteq, \cup, \cap)$ and $(A^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp)$.



$\gamma(\alpha((P))$
P
$\gamma(P^\sharp)$
$\gamma(A^\sharp)$
$\mathcal{A}$
$\alpha(P)$
$P^\sharp$
$\alpha(\gamma(P^\sharp))$
$\alpha(A)$

Conclusion: what is a *good* approximation space ?

1. concrete world: a complete lattice, generally of the form $(\mathcal{P}(\mathcal{D}), \subseteq, \bigcup, \bigcap)$
2. abstract world: a complete lattice $\left(A^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bigsqcap^\sharp\right)$
3. link between them: Galois connection.

$$\left(\mathcal{P}(\mathcal{D}), \subseteq, \bigcup, \bigcap\right) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp)$$

$$\begin{aligned} &a^\sharp \in A^\sharp \textit{ is a correct approximation of } a \in \mathcal{P}(\mathcal{D}) \\ &\iff \alpha(a) \sqsubseteq^\sharp a^\sharp \\ &\iff a \subseteq \gamma(a^\sharp) \end{aligned}$$

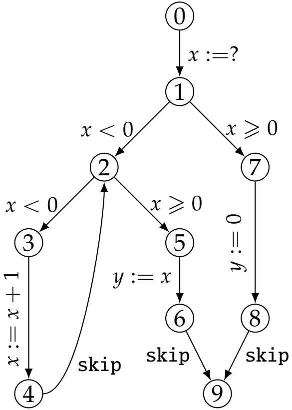
α: abstraction function $\qquad$ γ: concretisation function

Remark : in practice, γ is sufficient to prove the soundness of analyses, but we lose some *nice* theorems...

Just put some $^\sharp$...

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

control flow graph



collecting semantics

$$
\begin{array}{lcl}
X_0 & = & Env \\
X_1 & = & [\![x :=?]\!]\,(X_0) \\
X_2 & = & [\![x < 0]\!]\,(X_1) \cup X_4 \\
X_3 & = & [\![x < 0]\!]\,(X_2) \\
X_4 & = & [\![x := x + 1]\!]\,(X_3) \\
X_5 & = & [\![x \geqslant 0]\!]\,(X_2) \\
X_6 & = & [\![y := x]\!]\,(X_5) \\
X_7 & = & [\![x \geqslant 0]\!]\,(X_1) \\
X_8 & = & [\![y := 0]\!]\,(X_7) \\
X_9 & = & X_6 \cup X_8
\end{array}
$$

abstract semantics

$$
\begin{array}{lcl}
X_0^\sharp & = & \top_{Env}^\sharp \\
X_1^\sharp & = & [\![x :=?]\!]^\sharp\,(X_0^\sharp) \\
X_2^\sharp & = & [\![x < 0]\!]^\sharp\,(X_1^\sharp) \sqcup^\sharp X_4^\sharp \\
X_3^\sharp & = & [\![x < 0]\!]^\sharp\,(X_2^\sharp) \\
X_4^\sharp & = & [\![x := x + 1]\!]^\sharp\,(X_3^\sharp) \\
X_5^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_2^\sharp) \\
X_6^\sharp & = & [\![y := x]\!]^\sharp\,(X_5^\sharp) \\
X_7^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_1^\sharp) \\
X_8^\sharp & = & [\![y := 0]\!]^\sharp\,(X_7^\sharp) \\
X_9^\sharp & = & X_6^\sharp \sqcup^\sharp X_8^\sharp
\end{array}
$$

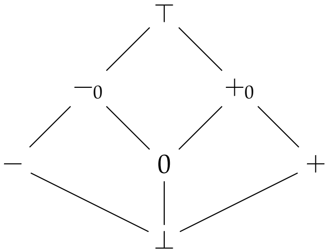
Abstract semantics : the ingredients

- A lattice structure $(Env^\sharp, \sqsubseteq^\sharp_{Env}, \sqcup^\sharp_{Env}, \sqcap^\sharp_{Env}, \perp^\sharp_{Env}, \top^\sharp_{Env})$
  - $\sqsubseteq^\sharp_{Env}$ is an approximation of $\subseteq$
  - $\sqcup^\sharp_{Env}$ is an approximation of $\cup$
  - $\sqcap^\sharp_{Env}$ is an approximation of $\cap$
  - $\perp^\sharp_{Env}$ is an approximation of $\emptyset$
  - $\top^\sharp_{Env}$ is an approximation of Env
- For all $x \in \mathbb{V}$, $e \in Exp$, $[\![x := e]\!]^\sharp \in Env^\sharp \to Env^\sharp$ an approximation of $[\![x := e]\!]$
- For all $t \in test$, $[\![t]\!]^\sharp \in Env^\sharp \to Env^\sharp$ an approximation of $[\![t]\!]$.

Ocaml code of the generic analyser

```
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



$\bot$	represents the property	$\emptyset$
$-$	represents the property	$\{\, z \mid z < 0 \,\}$
0	represents the property	$\{0\}$
$+$	represents the property	$\{\, z \mid z > 0 \,\}$
$-_0$	represents the property	$\{\, z \mid z \leqslant 0 \,\}$
$+_0$	represents the property	$\{\, z \mid z \geqslant 0 \,\}$
$\top$	represents the property	$\mathbb{Z}$

$\boldsymbol{Env}^\sharp \stackrel{\text{def}}{=} \mathbb{V} \rightarrow \boldsymbol{Sign}$: a sign is associated to each variable.

An abstraction by signs : example

$$
\begin{array}{rcl}
X_0^\sharp & = & \top_{Env}^\sharp \\
X_1^\sharp & = & [\![x := ?]\!]^\sharp\,(X_0^\sharp) \\
X_2^\sharp & = & [\![x < 0]\!]^\sharp\,(X_1^\sharp) \sqcup^\sharp X_4^\sharp \\
X_3^\sharp & = & [\![x < 0]\!]^\sharp\,(X_2^\sharp) \\
X_4^\sharp & = & [\![x := x + 1]\!]^\sharp\,(X_3^\sharp) \\
X_5^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_2^\sharp) \\
X_6^\sharp & = & [\![y := x]\!]^\sharp\,(X_5^\sharp) \\
X_7^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_1^\sharp) \\
X_8^\sharp & = & [\![y := 0]\!]^\sharp\,(X_7^\sharp) \\
X_9^\sharp & = & X_6^\sharp \sqcup^\sharp X_8^\sharp
\end{array}
\quad \xrightarrow[\text{simplifies into}]{\text{which}} \quad
\begin{array}{rcl}
X_0^\sharp & = & [x : \top;\, y : \top] \\
X_1^\sharp & = & X_0^\sharp[x \mapsto \top] \\
X_2^\sharp & = & X_1^\sharp[x \mapsto -] \sqcup^\sharp X_4^\sharp \\
X_3^\sharp & = & X_2^\sharp[x \mapsto -] \\
X_4^\sharp & = & X_3^\sharp[x \mapsto \mathrm{succ}^\sharp(X_3^\sharp(x))] \\
X_5^\sharp & = & X_2^\sharp[x \mapsto +_0] \\
X_6^\sharp & = & X_5^\sharp[y \mapsto X_5^\sharp(x)] \\
X_7^\sharp & = & X_1^\sharp[x \mapsto +_0] \\
X_8^\sharp & = & X_7^\sharp[y \mapsto 0] \\
X_9^\sharp & = & X_6^\sharp \sqcup^\sharp X_8^\sharp
\end{array}
$$

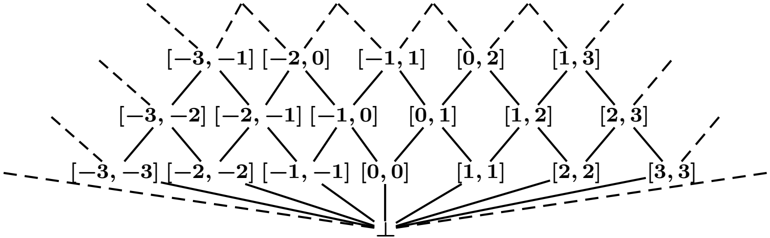
with

$$
\begin{array}{rcl}
\mathrm{succ}^\sharp(\bot) & = & \bot \\
\mathrm{succ}^\sharp(-) & = & -_0 \\
\mathrm{succ}^\sharp(0) & = & \mathrm{succ}^\sharp(+) = \mathrm{succ}^\sharp(+_0) = + \\
\mathrm{succ}^\sharp(-_0) & = & \mathrm{succ}^\sharp(\top) = \top
\end{array}
$$

Abstraction by intervals

$$Int \stackrel{\text{def}}{=} \{\, [a,b] \mid a,b \in \overline{\mathbb{Z}},\ a \leqslant b \,\} \cup \{\bot\}$$

with $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$.
$\bot$ represents $\emptyset$ and $[a,b]$ the property $\{z \mid a \leqslant z \leqslant b\}$.



$\boldsymbol{Env^\sharp} \stackrel{\text{def}}{=} \mathbb{V} \to \boldsymbol{Int}$: an interval is associated to each variable.

Abstraction by intervals : example

$$\begin{array}{rcl}
X_0^\sharp & = & \top_{Env}^\sharp \\
X_1^\sharp & = & [\![x := ?]\!]^\sharp\,(X_0^\sharp) \\
X_2^\sharp & = & [\![x < 0]\!]^\sharp\,(X_1^\sharp) \sqcup^\sharp X_4^\sharp \\
X_3^\sharp & = & [\![x < 0]\!]^\sharp\,(X_2^\sharp) \\
X_4^\sharp & = & [\![x := x + 1]\!]^\sharp\,(X_3^\sharp) \\
X_5^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_2^\sharp) \\
X_6^\sharp & = & [\![y := x]\!]^\sharp\,(X_5^\sharp) \\
X_7^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_1^\sharp) \\
X_8^\sharp & = & [\![y := 0]\!]^\sharp\,(X_7^\sharp) \\
X_9^\sharp & = & X_6^\sharp \sqcup^\sharp X_8^\sharp
\end{array}$$

$$\begin{array}{rcl}
X_0^\sharp & = & [x : [-\infty, +\infty];\ y : [-\infty, +\infty]] \\
X_1^\sharp & = & X_0^\sharp[x \mapsto [-\infty, +\infty]] \\
X_2^\sharp & = & X_1^\sharp[x \mapsto X_1^\sharp(x) \sqcap^\sharp [-\infty, -1]] \sqcup^\sharp X_4^\sharp \\
X_3^\sharp & = & X_2^\sharp[x \mapsto X_2^\sharp(x) \sqcap^\sharp [-\infty, -1]] \\
X_4^\sharp & = & X_3^\sharp[x \mapsto \mathrm{succ}^\sharp(X_3^\sharp(x))] \\
X_5^\sharp & = & X_2^\sharp[x \mapsto X_2^\sharp(x) \sqcap^\sharp [0, +\infty]] \\
X_6^\sharp & = & X_5^\sharp[y \mapsto X_5^\sharp(x)] \\
X_7^\sharp & = & X_1^\sharp[x \mapsto X_1^\sharp(x) \sqcap^\sharp [0, +\infty]] \\
X_8^\sharp & = & X_7^\sharp[y \mapsto [0, 0]] \\
X_9^\sharp & = & X_6^\sharp \sqcup^\sharp X_8^\sharp
\end{array}$$

with

$$\begin{array}{rcl}
\mathrm{succ}^\sharp(\bot) & = & \bot \\
\mathrm{succ}^\sharp([a, b]) & = & [a + 1, b + 1]
\end{array}$$

Demo

A small While analyzer is available (sign and interval analysis).

`http://sawd.irisa.fr/scripts/while/`

Abstraction by intervals

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

Lattice :

$$\frac{I \in \mathrm{Int}}{\bot \sqsubseteq_{\mathrm{Int}} I} \qquad \frac{c \leqslant a \qquad b \leqslant d \qquad a,b,c,d \in \overline{\mathbb{Z}}}{[a,b] \sqsubseteq_{\mathrm{Int}} [c,d]}$$

$$\begin{array}{rcl} I \sqcup_{\mathrm{Int}} \bot & \stackrel{\text{def}}{=} & I,\ \forall I \in \mathrm{Int} \\ \bot \sqcup_{\mathrm{Int}} I & \stackrel{\text{def}}{=} & I,\ \forall I \in \mathrm{Int} \\ [a,b] \sqcup_{\mathrm{Int}} [c,d] & \stackrel{\text{def}}{=} & [\min(a,c), \max(b,d)] \end{array}$$

$$\begin{array}{rcl} I \sqcap_{\mathrm{Int}} \bot & \stackrel{\text{def}}{=} & \bot,\ \forall I \in \mathrm{Int} \\ \bot \sqcap_{\mathrm{Int}} I & \stackrel{\text{def}}{=} & \bot,\ \forall I \in \mathrm{Int} \\ [a,b] \sqcap_{\mathrm{Int}} [c,d] & \stackrel{\text{def}}{=} & \rho_{\mathrm{Int}}([\max(a,c), \min(b,d)]) \end{array}$$

with $\rho_{\mathrm{Int}} \in (\overline{\mathbb{Z}} \times \overline{\mathbb{Z}}) \to \mathrm{Int}$ defined by

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

$$\begin{aligned} \bot_{\mathrm{Int}} &\stackrel{\text{def}}{=} \bot \\ \top_{\mathrm{Int}} &\stackrel{\text{def}}{=} [-\infty, +\infty] \end{aligned}$$

$$\begin{aligned} \alpha_{\mathrm{Int}}(S) &\stackrel{\text{def}}{=} \bot && \text{if } S = \emptyset \\ \alpha_{\mathrm{Int}}(S) &\stackrel{\text{def}}{=} [\min(S), \max(S)] && \text{otherwise} \end{aligned}$$

$$\begin{aligned} \gamma_{\mathrm{Int}}(\bot) &\stackrel{\text{def}}{=} \emptyset \\ \gamma_{\mathrm{Int}}([a, b]) &\stackrel{\text{def}}{=} \{ z \in \mathbb{Z} \mid a \leqslant z \text{ and } z \leqslant b \} \end{aligned}$$

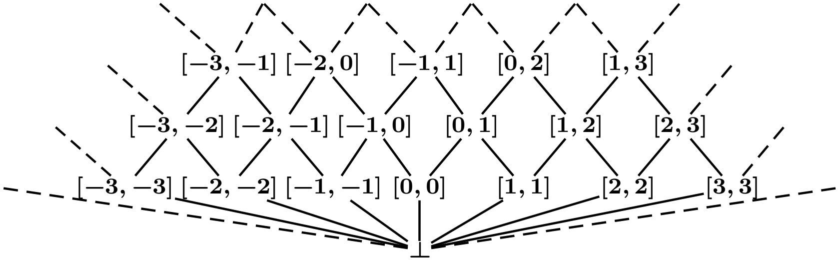
All the other operators are *strict*: they return $\bot$ if one of their arguments is $\bot$.

$$
\begin{array}{rcl}
+^\sharp([a,b],[c,d]) & = & [a+c, b+d] \\
-^\sharp([a,b],[c,d]) & = & [a-d, b-c] \\
\times^\sharp([a,b],[c,d]) & = & [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)]
\end{array}
$$

$$
\begin{array}{rcl}
[\![+]\!]\!\downarrow^\sharp_{\text{op}}([a,b],[c,d],[e,f]) & = & (\rho(\max(c,a-f),\min(d,b-e)), \\
 & & \quad \rho(\max(e,a-d),\min(f,b-c))) \\
[\![-]\!]\!\downarrow^\sharp_{\text{op}}([a,b],[c,d],[e,f]) & = & (\rho(\max(c,a+e),\min(d,b+f)), \\
 & & \rho(\max(e,c-b),\min(f,d-a))) \\
[\![*]\!]\!\downarrow^\sharp_{\text{op}}([a,b],[c,d],[e,f]) & = & ([c,d],[e,f])
\end{array}
$$

$$
\begin{array}{rcl}
[\![=]\!]\!\downarrow^\sharp_{\text{comp}}([a,b],[c,d]) & = & ([a,b]\sqcap_{\text{Int}}[c,d], [a,b]\sqcap_{\text{Int}}[c,d]) \\
[\![<]\!]\!\downarrow^\sharp_{\text{comp}}([a,b],[c,d]) & = & ([a,b]\sqcap_{\text{Int}}[-\infty,d-1], [a+1,+\infty]\sqcap_{\text{Int}}[c,d]) \\
[\![\leqslant]\!]\!\downarrow^\sharp_{\text{comp}}([a,b],[c,d]) & = & ([a,b]\sqcap_{\text{Int}}[-\infty,d], [a,+\infty]\sqcap_{\text{Int}}[c,d]) \\
[\![\neq]\!]\!\downarrow^\sharp_{\text{comp}}([a,b],[c,d]) & = & ?\ \textit{exercise...} \\
\text{const}(n)^\sharp & = & [n,n]
\end{array}
$$

Convergence problem



Such a lattice does not satisfy the ascending chain condition.

Example of infinite increasing chain :

$$\perp \sqsubset [0,0] \sqsubset [0,1] \sqsubset \cdots \sqsubset [0,n] \sqsubset \cdots$$

Solution : dynamic approximation

Fixpoint 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* $\mathrm{lfp}(f) \sqsubseteq a$.

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

- The lattice does not satisfies 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

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

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 ∇ operator that is independent of F.

Widening : definition

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

- $\forall x, x' \in L, x \sqcup x' \sqsubseteq x \nabla x'$ (implies (i) & (ii))
- 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 $x^0 = \bot, x^{n+1} = F(x^n)$
by $y^0 = \bot, y^{n+1} = y^n \nabla F(y^n)$

Widening: theorem

Theorem

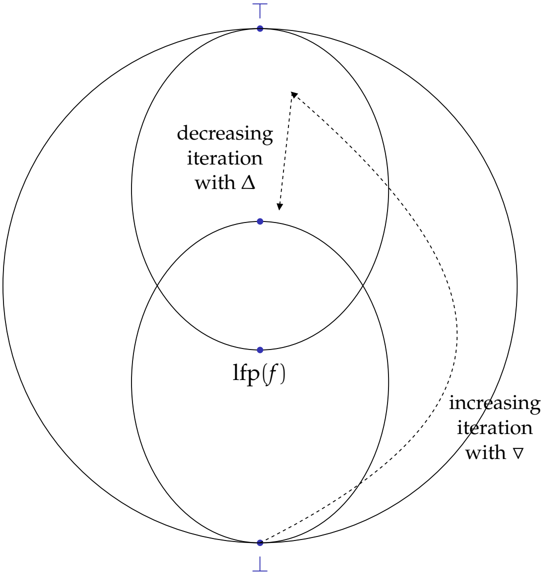
Let

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

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

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

Scheme


$\top$
decreasing
iteration
with Δ
$\mathrm{lfp}(f)$
increasing
iteration
with ∇
$\bot$

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:

$$
\begin{array}{lcl}
[a,b]\nabla_{\mathrm{Int}}[a',b'] & = & [\ \ \text{if } a' < a \text{ then } -\infty \text{ else } a, \\
 & & \ \ \ \ \text{if } b' > b \text{ then } +\infty \text{ else } b\,] \\
\bot\nabla_{\mathrm{Int}}[a',b'] & = & [a',b'] \\
I\ \nabla_{\mathrm{Int}}\ \bot & = & I
\end{array}
$$

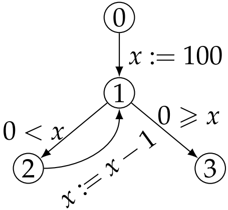
Examples:

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

$[-3,4]\nabla_{\mathrm{Int}}[-3,5] = [-3,+\infty]$

Example

```
x := 100;
while 0 < x {
  x := x − 1;
}
```



$$X_1 = [100, 100] \sqcup_{\mathrm{Int}} \left(X_2 -^{\sharp} [1, 1]\right)$$

$$X_2 = [1, +\infty] \sqcap_{\mathrm{Int}} X_1$$

$$X_3 = [-\infty, 0] \sqcap_{\mathrm{Int}} X_1$$

Example : without widening

$$
\begin{aligned}
X_1 &= [100, 100] \sqcup_{\text{Int}} \left(X_2 -^{\sharp} [1, 1]\right) \\
X_2 &= [1, +\infty] \sqcap_{\text{Int}} X_1 \\
X_3 &= [-\infty, 0] \sqcap_{\text{Int}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = \bot & X_1^{n+1} = [100, 100] \sqcup_{\text{Int}} \left(X_2^n -^{\sharp} [1, 1]\right) \\
X_2^0 = \bot & X_2^{n+1} = [1, +\infty] \sqcap_{\text{Int}} X_1^{n+1} \\
X_3^0 = \bot & X_3^{n+1} = [-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1}
\end{array}
$$

X_1	$\bot$	$\cdots$
X_2	$\bot$	$\cdots$
X_3	$\bot$	$\cdots$

Example : without widening

$$
\begin{aligned}
X_1 &= [100, 100] \sqcup_{\text{Int}} \left(X_2 -^{\sharp} [1, 1]\right) \\
X_2 &= [1, +\infty] \sqcap_{\text{Int}} X_1 \\
X_3 &= [-\infty, 0] \sqcap_{\text{Int}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = \bot & X_1^{n+1} = [100, 100] \sqcup_{\text{Int}} \left(X_2^n -^{\sharp} [1, 1]\right) \\
X_2^0 = \bot & X_2^{n+1} = [1, +\infty] \sqcap_{\text{Int}} X_1^{n+1} \\
X_3^0 = \bot & X_3^{n+1} = [-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1}
\end{array}
$$

X_1	$\bot$	$[100, 100]$	$[99, 100]$	$[98, 100]$	$[97, 100]$	$\cdots$	$[1, 100]$	$[0, 100]$
X_2	$\bot$	$[100, 100]$	$[99, 100]$	$[98, 100]$	$[97, 100]$	$\cdots$	$[1, 100]$	$[1, 100]$
X_3	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$	$\bot$	$[0, 0]$

Example : with widening at each nodes of the cfg

$$
\begin{aligned}
X_1 &= [100,100] \sqcup_{\text{Int}} \left(X_2 -^{\sharp} [1,1]\right) \\
X_2 &= [1,+\infty] \sqcap_{\text{Int}} X_1 \\
X_3 &= [-\infty,0] \sqcap_{\text{Int}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = \bot & X_1^{n+1} = X_1^n \nabla_{\text{Int}} \left([100,100] \sqcup_{\text{Int}} \left(X_2^n -^{\sharp} [1,1]\right)\right) \\
X_2^0 = \bot & X_2^{n+1} = X_2^n \nabla_{\text{Int}} \left([1,+\infty] \sqcap_{\text{Int}} X_1^{n+1}\right) \\
X_3^0 = \bot & X_3^{n+1} = X_3^n \nabla_{\text{Int}} \left([-\infty,0] \sqcap_{\text{Int}} X_1^{n+1}\right)
\end{array}
$$

X_1	$\bot$
X_2	$\bot$
X_3	$\bot$

Example : with widening at each nodes of the cfg

$$\begin{aligned} X_1 &= [100,100] \sqcup_{\text{Int}} \left(X_2 -^\sharp [1,1]\right) \\ X_2 &= [1,+\infty] \sqcap_{\text{Int}} X_1 \\ X_3 &= [-\infty,0] \sqcap_{\text{Int}} X_1 \end{aligned}$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$\begin{array}{ll} X_1^0 = \bot & X_1^{n+1} = X_1^n \nabla_{\text{Int}} \left([100,100] \sqcup_{\text{Int}} \left(X_2^n -^\sharp [1,1]\right)\right) \\ X_2^0 = \bot & X_2^{n+1} = X_2^n \nabla_{\text{Int}} \left([1,+\infty] \sqcap_{\text{Int}} X_1^{n+1}\right) \\ X_3^0 = \bot & X_3^{n+1} = X_3^n \nabla_{\text{Int}} \left([-\infty,0] \sqcap_{\text{Int}} X_1^{n+1}\right) \end{array}$$

X_1	$\bot$	$[100,100]$	$[-\infty,100]$
X_2	$\bot$	$[100,100]$	$[-\infty,100]$
X_3	$\bot$	$\bot$	$[-\infty,0]$

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 $\begin{cases} x_0 & = & a \\ x_{k+1} & = & f(x_k) \end{cases}$ admits for limit ($\bigsqcap\{x_n\}$) the greatest fixpoint of f lower than a (written $\mathrm{gfp}_a(f)$). In particular, $\mathrm{lfp}(f) \sqsubseteq \bigsqcap\{x_n\}$. Each intermediate step is a correct approximation:

$$\forall k,\ \mathrm{lfp}(f) \sqsubseteq \mathrm{gfp}_a(f) \sqsubseteq x_k \sqsubseteq a$$

Narrowing : definition

A *narrowing* is an operator $\Delta : L \times L \rightarrow L$ 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.

Narrowing: decreasing iteration

Theorem

If Δ 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 $\begin{cases} x_0 & = & a \\ x_{k+1} & = & x_k \Delta f(x_k) \end{cases}$ *stabilizes after a finite number of steps on a post-fixpoint of f lower than a.*

Narrowing on intervals

$$
\begin{array}{rcl}
[a,b]\Delta_{\mathrm{Int}}[c,d] & = & [\text{if } a = -\infty \text{ then } c \text{ else } a \;\; ; \;\; \text{if } b = +\infty \text{ then } d \text{ else } b] \\
I \;\; \Delta_{\mathrm{Int}} \;\; \bot & = & \bot \\
\bot \;\; \Delta_{\mathrm{Int}} \;\; I & = & \bot
\end{array}
$$

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

$$\begin{array}{l} X_1 = [100,100] \sqcup_{\mathrm{Int}} \left(X_2 -^\sharp [1,1]\right) \\ X_2 = [1,+\infty] \sqcap_{\mathrm{Int}} X_1 \\ X_3 = [-\infty,0] \sqcap_{\mathrm{Int}} X_1 \end{array}$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$\begin{array}{ll} X_1^0 = [-\infty,100] & X_1^{n+1} = X_1^n \Delta_{\mathrm{Int}} \left([100,100] \sqcup_{\mathrm{Int}} \left(X_2^n -^\sharp [1,1]\right)\right) \\ X_2^0 = [-\infty,100] & X_2^{n+1} = X_2^n \Delta_{\mathrm{Int}} \left([1,+\infty] \sqcap_{\mathrm{Int}} X_1^{n+1}\right) \\ X_3^0 = [-\infty,0] & X_3^{n+1} = X_3^n \Delta_{\mathrm{Int}} \left([-\infty,0] \sqcap_{\mathrm{Int}} X_1^{n+1}\right) \end{array}$$

X_1	$[-\infty,100]$
X_2	$[-\infty,100]$
X_3	$[-\infty,0]$

Example : with narrowing at each node of the cfg

$$
\begin{aligned}
X_1 &= [100, 100] \sqcup_{\text{Int}} \left(X_2 -^{\sharp} [1, 1]\right) \\
X_2 &= [1, +\infty] \sqcap_{\text{Int}} X_1 \\
X_3 &= [-\infty, 0] \sqcap_{\text{Int}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = [-\infty, 100] & X_1^{n+1} = X_1^n \Delta_{\text{Int}} \left([100, 100] \sqcup_{\text{Int}} \left(X_2^n -^{\sharp} [1, 1]\right)\right) \\
X_2^0 = [-\infty, 100] & X_2^{n+1} = X_2^n \Delta_{\text{Int}} \left([1, +\infty] \sqcap_{\text{Int}} X_1^{n+1}\right) \\
X_3^0 = [-\infty, 0] & X_3^{n+1} = X_3^n \Delta_{\text{Int}} \left([-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1}\right)
\end{array}
$$

X_1	$[-\infty, 100]$	$[-\infty, 100]$	$[0, 100]$
X_2	$[-\infty, 100]$	$[1, 100]$	$[1, 100]$
X_3	$[-\infty, 0]$	$[-\infty, 0]$	$[0, 0]$

The particular case of an equation system

Consider a system

$$\begin{cases} x_1 & = & f_1(x_1, \dots, x_n) \\ & \vdots & \\ x_n & = & f_n(x_1, \dots, x_n) \end{cases}$$

with $f_1, \dots, f_n$ monotones.

Standard iteration:

$$\begin{array}{rcl} x_1^{i+1} & = & f_1(x_1^i, \dots, x_n^i) \\ x_2^{i+1} & = & f_2(x_1^i, \dots, x_n^i) \\ & \vdots & \\ x_n^{i+1} & = & f_n(x_1^i, \dots, x_n^i) \end{array}$$

Standard iteration with widening:

$$\begin{array}{rcl} x_1^{i+1} & = & x_1^i \nabla f_1(x_1^i, \dots, x_n^i) \\ x_2^{i+1} & = & x_2^i \nabla f_2(x_1^i, \dots, x_n^i) \\ & \vdots & \\ x_n^{i+1} & = & x_n^i \nabla f_n(x_1^i, \dots, x_n^i) \end{array}$$

The particular case of an equation system

$$\left\{\begin{array}{ccc} x_1 & = & f_1(x_1,\ldots,x_n) \\ & \vdots & \\ x_n & = & f_n(x_1,\ldots,x_n) \end{array}\right.$$

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

$$\forall k = 1..n,\ x_k^{i+1} = \begin{array}{ll} x_k^i \triangledown f_k(x_1^i,\ldots,x_n^i) & \text{if } k \in W \\ f_k(x_1^i,\ldots,x_n^i) & \text{otherwise} \end{array}$$

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.

Patrick Cousot

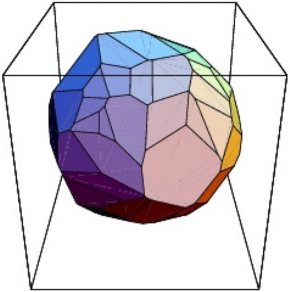
Nicolas Halbwachs

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

Convex polyhedra

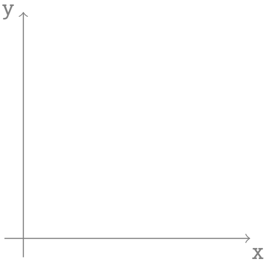
A convex polyhedron can be defined algebraically as the set of solutions of a system of linear inequalities.

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



Polyhedral analysis

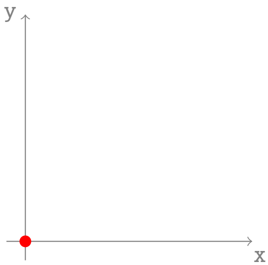
State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.




```
x = 0; y = 0;


while (x<6) {
  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;
```

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

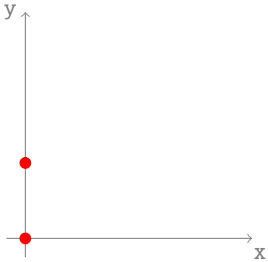
```
while (x<6) {
  if (?) {
```

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

```
    y = y+2;

  };


  x = x+1;

}
```

Polyhedral analysis

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



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

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

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

```
while (x<6) {
  if (?) {
```

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

```
    y = y+2;
```

$\{x = 0 \wedge y = 2\}$

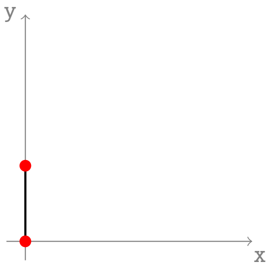
```
  };
```

$\{x = 0 \wedge y = 0\} \uplus \{x = 0 \wedge y = 2\}$

```
  x = x+1;

}
```

Polyhedral analysis

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



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

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

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

```
while (x<6) {
  if (?) {
```

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

```
    y = y+2;
```

$\{x = 0 \ \wedge \ y = 2\}$

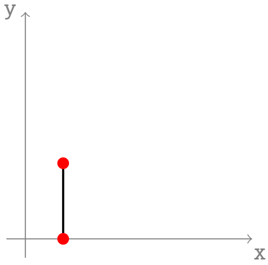
```
  };
```

$\{x = 0 \ \wedge \ 0 \leqslant y \leqslant 2\}$

```
  x = x+1;

}
```

Polyhedral analysis

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




```
x = 0; y = 0;
```
$\{x = 0 \ \wedge \ y = 0\}$

```
while (x<6) {
  if (?) {
```
$\{x = 0 \ \wedge \ y = 0\}$

```
    y = y+2;
```
$\{x = 0 \ \wedge \ y = 2\}$

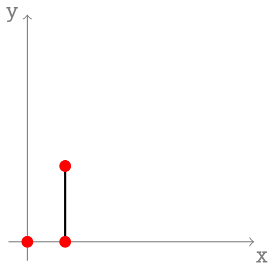
```
  };
```
$\{x = 0 \ \wedge \ 0 \leqslant y \leqslant 2\}$

```
  x = x+1;
```
$\{x = 1 \ \wedge \ 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




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

$\{x = 0 \wedge y = 0\} \uplus \{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
while (x<6) {
  if (?) {
```

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

```
    y = y+2;
```

$\{x = 0 \wedge y = 2\}$

```
  };
```

$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

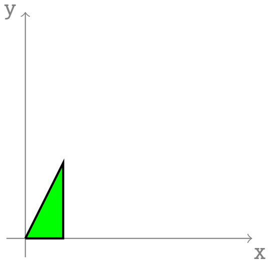
```
  x = x+1;
```

$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




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

$\{x \leqslant 1 \ \wedge\ 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```

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

```
    y = y+2;
```

$\{x = 0 \ \wedge\ y = 2\}$

```
  };
```

$\{x = 0 \ \wedge\ 0 \leqslant y \leqslant 2\}$

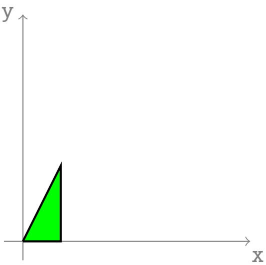
```
  x = x+1;
```

$\{x = 1 \ \wedge\ 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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



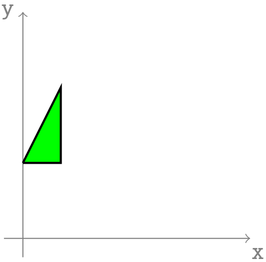

```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$
```
    y = y+2;
```
$\{x = 0 \wedge y = 2\}$
```
  };
```
$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$
```
}
```

Polyhedral analysis

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




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

$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```

$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```

$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x + 2\}$

```
  };
```

$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

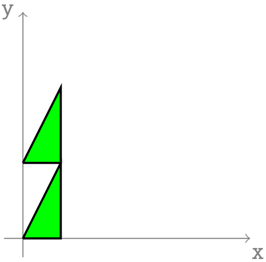
```
  x = x+1;
```

$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




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

$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```

$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```

$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$

```
  };
```

$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$
$\uplus\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$

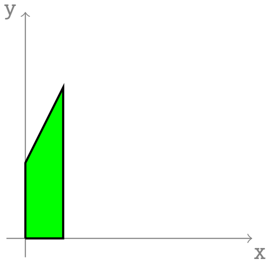
```
  x = x+1;
```

$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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



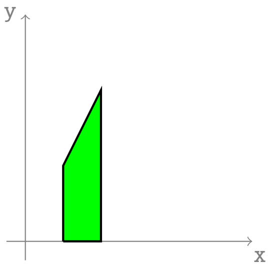

```
x = 0; y = 0;
```
$\{x \leqslant 1 \ \wedge \ 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \ \wedge \ 0 \leqslant y \leqslant 2x\}$
```
    y = y+2;
```
$\{x \leqslant 1 \ \wedge \ 2 \leqslant y \leqslant 2x + 2\}$
```
  };
```
$\{0 \leqslant x \leqslant 1 \ \wedge \ 0 \leqslant y \leqslant 2x + 2\}$

```
  x = x+1;
```
$\{x = 1 \ \wedge \ 0 \leqslant y \leqslant 2\}$
```
}
```

Polyhedral analysis

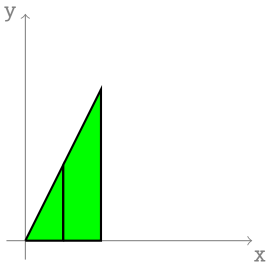
State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.




```
x = 0; y = 0;
        {x ⩽ 1 ∧ 0 ⩽ y ⩽ 2x}

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

  x = x+1;
        {1 ⩽ x ⩽ 2 ∧ 0 ⩽ y ⩽ 2x}
}
```

Polyhedral analysis

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

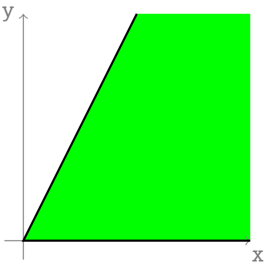


At loop headers, we use heuristics (widening) to ensure finite convergence.

```
x = 0; y = 0;
      {x ⩽ 1 ∧ 0 ⩽ y ⩽ 2x}
                  ∇ {x ⩽ 2 ∧ 0 ⩽ y ⩽ 2x}
while (x<6) {
  if (?) {
      {x ⩽ 1 ∧ 0 ⩽ y ⩽ 2x}
    y = y+2;
      {x ⩽ 1 ∧ 2 ⩽ y ⩽ 2x + 2}
  };
      {0 ⩽ x ⩽ 1 ∧ 0 ⩽ y ⩽ 2x + 2}

  x = x+1;
      {1 ⩽ x ⩽ 2 ∧ 0 ⩽ y ⩽ 2x}
}
```

Polyhedral analysis

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



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

```
x = 0; y = 0;
```
$\{0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$
```
    y = y+2;
```
$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$
```
  };
```
$\{0 \leqslant x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x+2\}$

```
  x = x+1;
```
$\{1 \leqslant x \leqslant 2 \wedge 0 \leqslant y \leqslant 2x\}$
```
}
```

Polyhedral analysis

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

By propagation we obtain a post-fixpoint

```
x = 0; y = 0;
        {0 ⩽ y ⩽ 2x}

while (x<6) {
  if (?) {
        {0 ⩽ y ⩽ 2x ∧ x ⩽ 5}
    y = y+2;
        {2 ⩽ y ⩽ 2x + 2 ∧ x ⩽ 5}
  };
        {0 ⩽ y ⩽ 2x + 2 ∧ 0 ⩽ x ⩽ 5}

  x = x+1;
        {0 ⩽ y ⩽ 2x ∧ 1 ⩽ x ⩽ 6}
}
        {0 ⩽ y ⩽ 2x ∧ 6 ⩽ x}
```

Polyhedral analysis

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

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

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

$\{0 \leqslant y \leqslant 2x \wedge x \leqslant 6\}$

```
while (x<6) {
  if (?) {
```

$\{0 \leqslant y \leqslant 2x \wedge x \leqslant 5\}$

```
    y = y+2;
```

$\{2 \leqslant y \leqslant 2x + 2 \wedge x \leqslant 5\}$

```
  };
```

$\{0 \leqslant y \leqslant 2x + 2 \wedge 0 \leqslant x \leqslant 5\}$

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

$\{0 \leqslant y \leqslant 2x \wedge 1 \leqslant x \leqslant 6\}$

```
}
```

$\{0 \leqslant y \leqslant 2x \wedge 6 = x\}$

Polyhedral analysis

A more complex example.

```
x = 0; y = A;
```

$$\{A \leqslant y \leqslant 2x + A \wedge x \leqslant N\}$$

```
while (x<N) {
  if (?) {
```

$$\{A \leqslant y \leqslant 2x + A \;\wedge\; x \leqslant N - 1\}$$

```
    y = y+2;
```

$$\{A + 2 \leqslant y \leqslant 2x + A + 2 \;\wedge\; x \leqslant N - 1\}$$

```
  };
```

$$\{A \leqslant y \leqslant 2x + A + 2 \;\wedge\; 0 \leqslant x \leqslant N - 1\}$$

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

$$\{A \leqslant y \leqslant 2x + A \;\wedge\; 1 \leqslant x \leqslant N\}$$

```
}
```

$$\{A \leqslant y \leqslant 2x + A \;\wedge\; N = x\}$$

The analysis accepts to replace some constants by parameters.

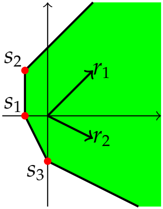
The four polyhedra operations

- $\uplus \in \mathbb{P}_n \times \mathbb{P}_n \to \mathbb{P}_n$: convex union
  - over-approximates the concrete union at junction points
- $\cap \in \mathbb{P}_n \times \mathbb{P}_n \to \mathbb{P}_n$: intersection
  - over-approximates the concrete intersection after a conditional intruction
- $[\![\mathtt{x} := e]\!] \in \mathbb{P}_n \to \mathbb{P}_n$: affine transformation
  - over-approximates the assignment of a variable by a linear expression
- $\nabla \in \mathbb{P}_n \times \mathbb{P}_n \to \mathbb{P}_n$: widening
  - ensures (and accelerates) convergence of (post-)fixpoint iteration
  - includes heuristics to infer loop invariants

```
x = 0; y = 0;
```
$$P_0 = [\![\mathtt{y} := \mathtt{0}]\!]\,[\![\mathtt{x} := \mathtt{0}]\!]\,(\mathbb{Q}^2) \ \nabla\ P_4$$
```
while (x<6) {
  if (?) {
```
$$P_1 = P_0 \cap \{x < 6\}$$
```
    y = y+2;
```
$$P_2 = [\![\mathtt{y} := \mathtt{y} + \mathtt{2}]\!]\,(P_1)$$
```
  };
```
$$P_3 = P_1 \uplus P_2$$
```
  x = x+1;
```
$$P_4 = [\![\mathtt{x} := \mathtt{x} + \mathtt{1}]\!]\,(P_3)$$
```
}
```
$$P_5 = P_0 \cap \{x \geqslant 6\}$$

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

$$P = \left\{ (x,y) \in \mathbb{Q}^2 \;\middle|\; \begin{array}{l} x \geqslant -1 \\ x - y \geqslant -3 \\ 2x + y \geqslant -2 \\ x + 2y \geqslant -4 \end{array} \right\}$$

  - by set of generators

$$P = \left\{ \lambda_1 s_1 + \lambda_2 s_2 + \lambda_3 s_3 + \mu_1 r_1 + \mu_2 r_2 \in \mathbb{Q}^2 \;\middle|\; \begin{array}{l} \lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2 \in \mathbb{R}^+ \\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{array} \right\}$$

  - 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

  P. Cousot and R. Cousot. Basic Concepts of Abstract Interpretation.
  `http://www.di.ens.fr/~cousot/COUSOTpapers/WCC04.shtml`

- technical but very complete (the logic programming part is optional) :

  P. Cousot and R. Cousot. Abstract Interpretation and Application to Logic Programs.
  `http://www.di.ens.fr/~cousot/COUSOTpapers/JLP92.shtml`

- a nice application of abstract interpretation theory to verify airbus flight commands

  P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE Analyser.
  `http://www.di.ens.fr/~cousot/COUSOTpapers/ESOP05.shtml`

References (2)

On the web:

- informal presentation of AI with nice pictures

  http://www.di.ens.fr/~cousot/AI/IntroAbsInt.html
- a short abstract of various works around AI

  http://www.di.ens.fr/~cousot/AI/
- very complete lecture notes

  http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/