# Validation par analyse statique

## Partie : Interprétation abstraite, cours 3/3

Pierre-Loïc Garoche

(merci à Pierre Roux pour ses contributions à ce cours)

ENAC

ENSEEIHT 2A

2022-2023

# Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes

# Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes
- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ directement en un $\mathcal{D}^\sharp$
  - *relationnel* : certaines combinaisons de $x$ et $y$ sont impossibles

# Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ directement en un $\mathcal{D}^\sharp$
  - *relationnel* : certaines combinaisons de $x$ et $y$ sont impossibles
  - $+$ plus précis
  - $-$ plus compliqué et plus coûteux

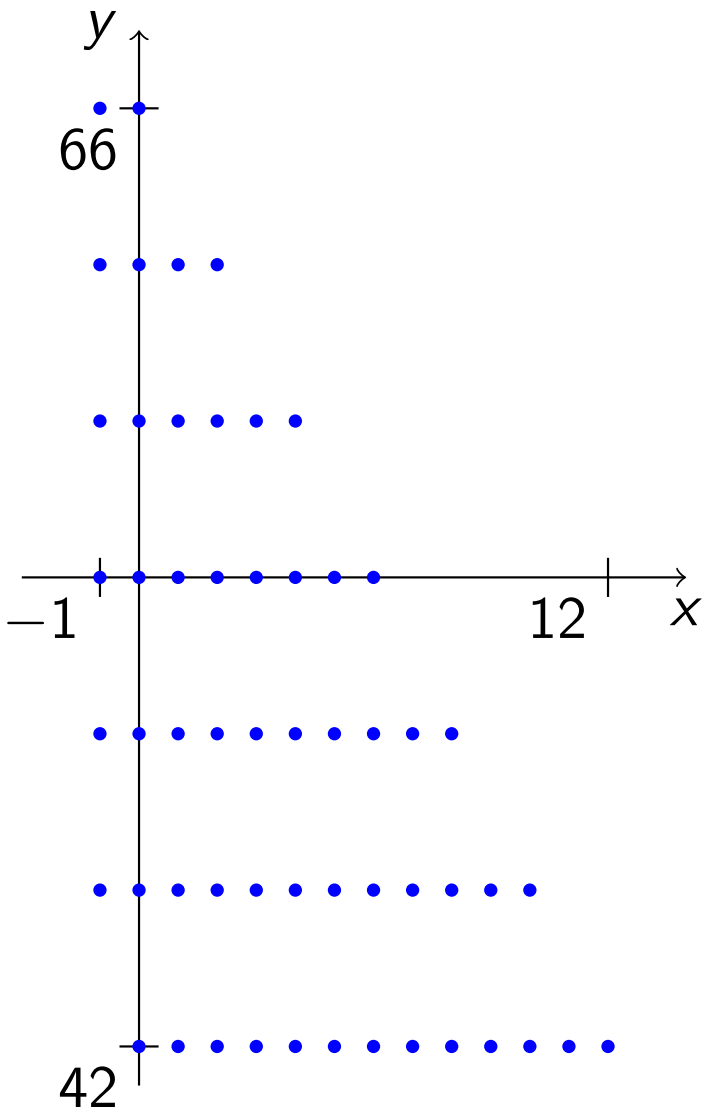
# Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes
  - le cours précédent
- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ directement en un $\mathcal{D}^\sharp$
  - *relationnel* : certaines combinaisons de $x$ et $y$ sont impossibles
  - $+$ plus précis
  - $-$ plus compliqué et plus coûteux
  - maintenant !

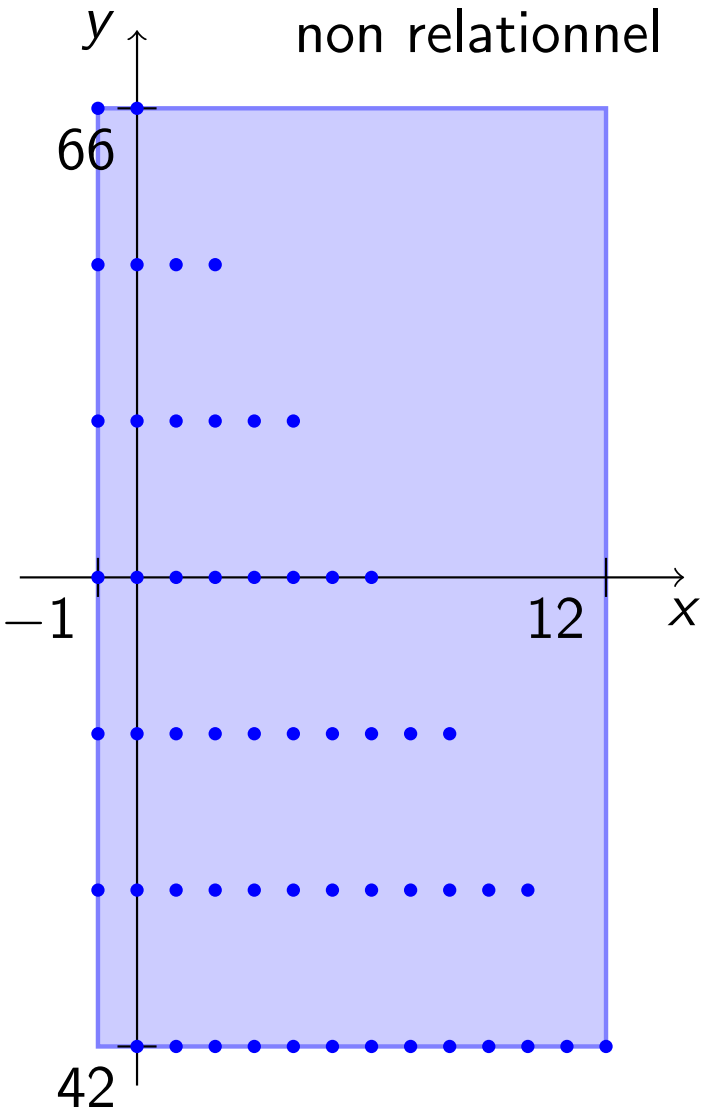
# Deux petits dessins valent mieux que de longs discours

Exemple précédent au point de programme 2 (invariant de boucle)



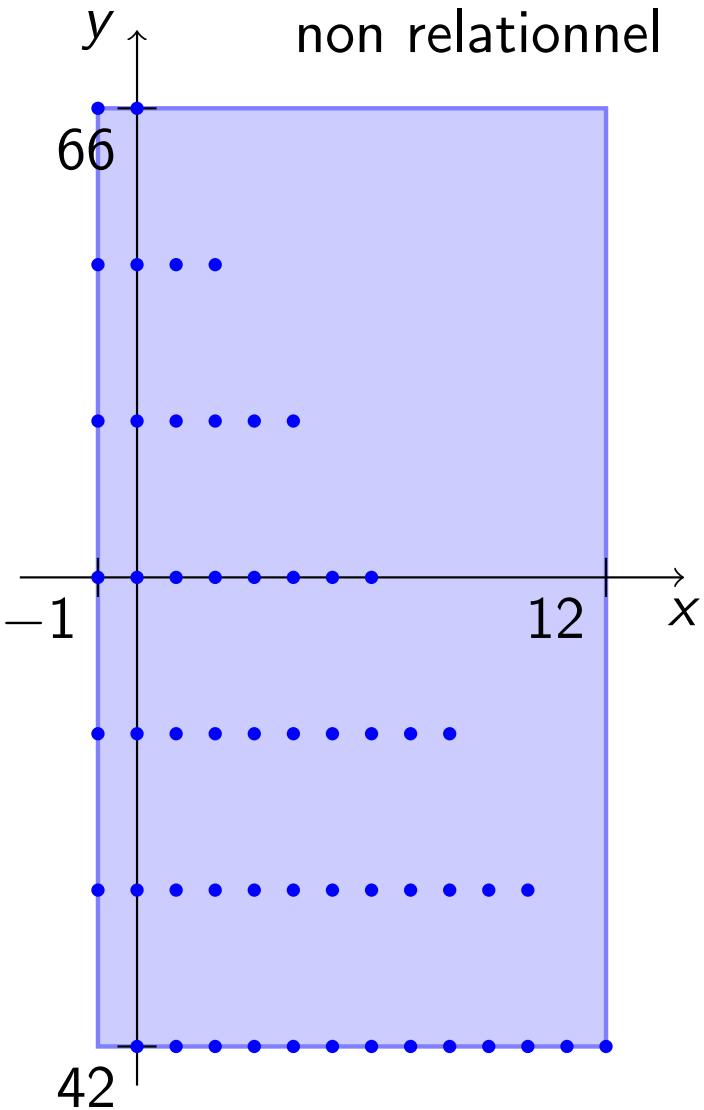
# Deux petits dessins valent mieux que de longs discours

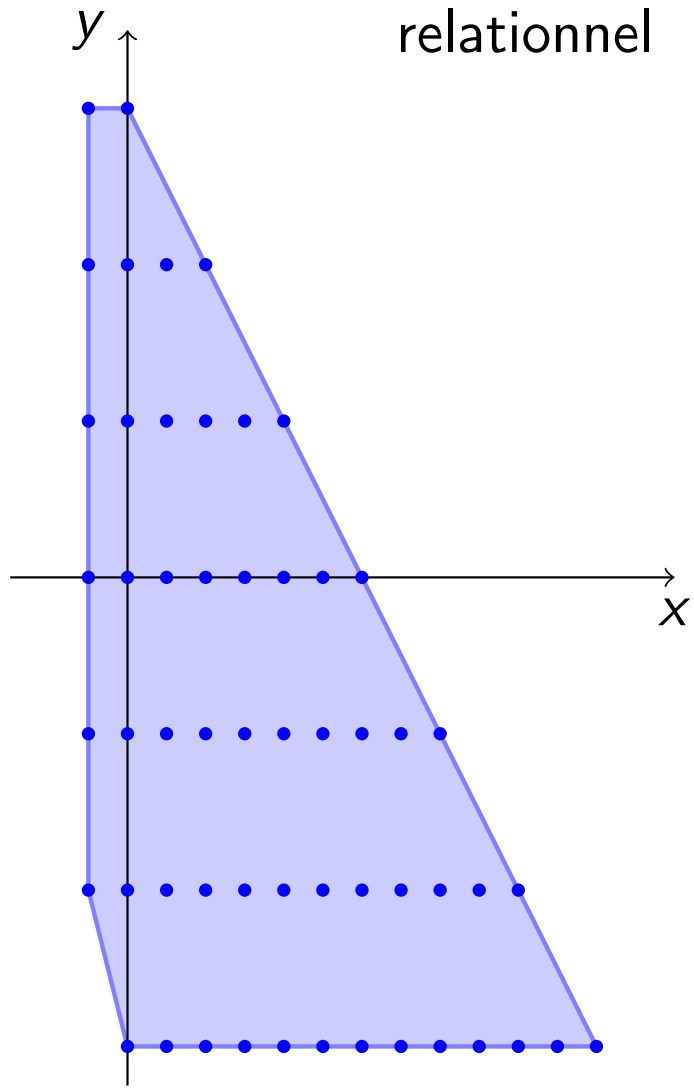
Exemple précédent au point de programme 2 (invariant de boucle)



# Deux petits dessins valent mieux que de longs discours

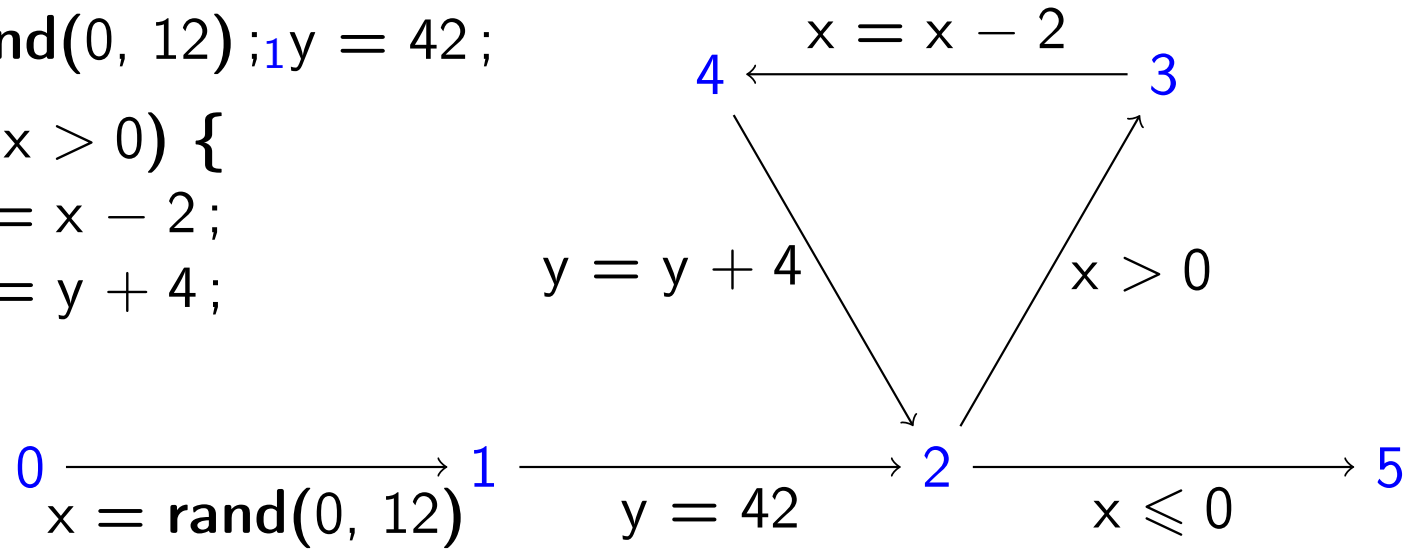
Exemple précédent au point de programme 2 (invariant de boucle)



# Limitations des domaines non relationnels

$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$
**while** $_2(x > 0)$ **{**
  $_3x = x - 2\,;$
  $_4y = y + 4\,;$
**}**$_5$



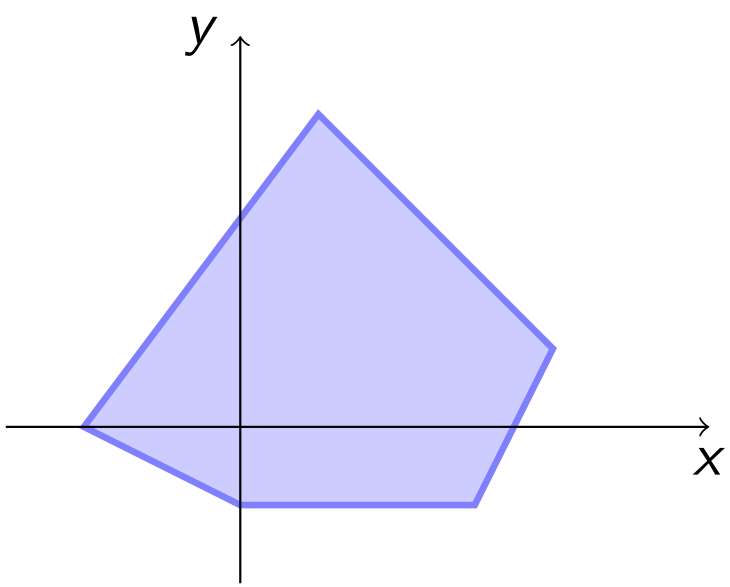

- Pour borner $y$, on a besoin de l'invariant $2x + y \leqslant 66$.
- Cet invariant de boucle ne peut être exprimé par aucun domaine non relationnel.

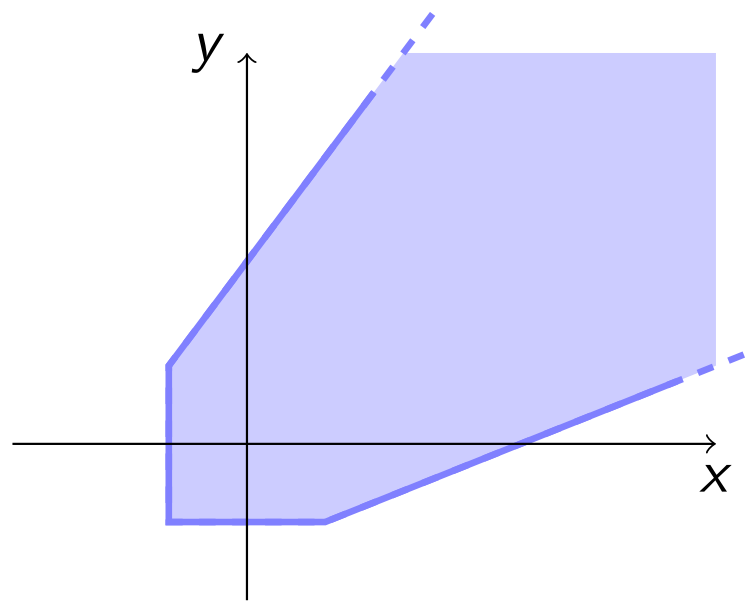
# Polyèdres

On s'intéresse aux polyèdres fermés convexes

soit des ensembles de la forme $\left\{ \rho \;\middle|\; \bigwedge_i \left( \sum_j a_{ij}\rho(v_j) \geqslant b_i \right) \right\}$

avec $a_{ij}, b_i \in \mathbb{Z}$ et $v_i \in \mathbb{V}$.

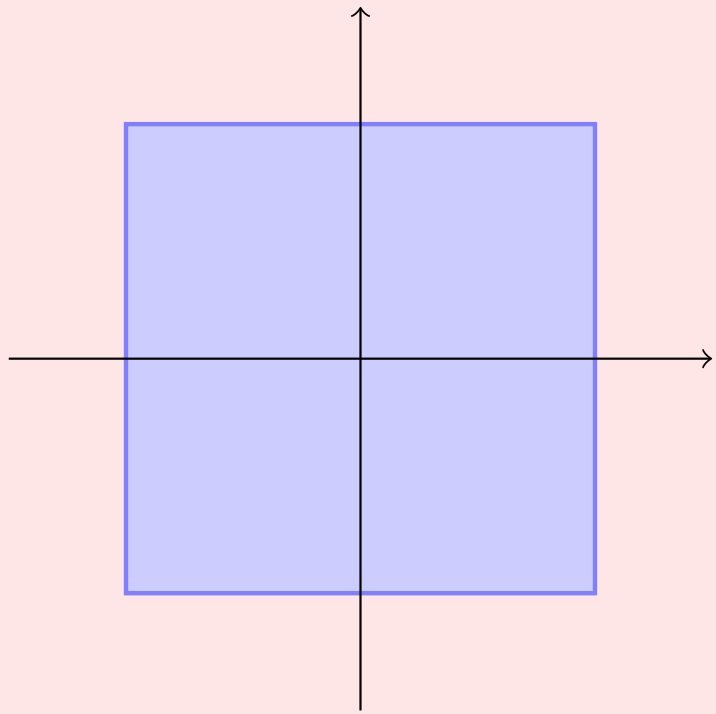


# Polyèdre, treillis

## Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.

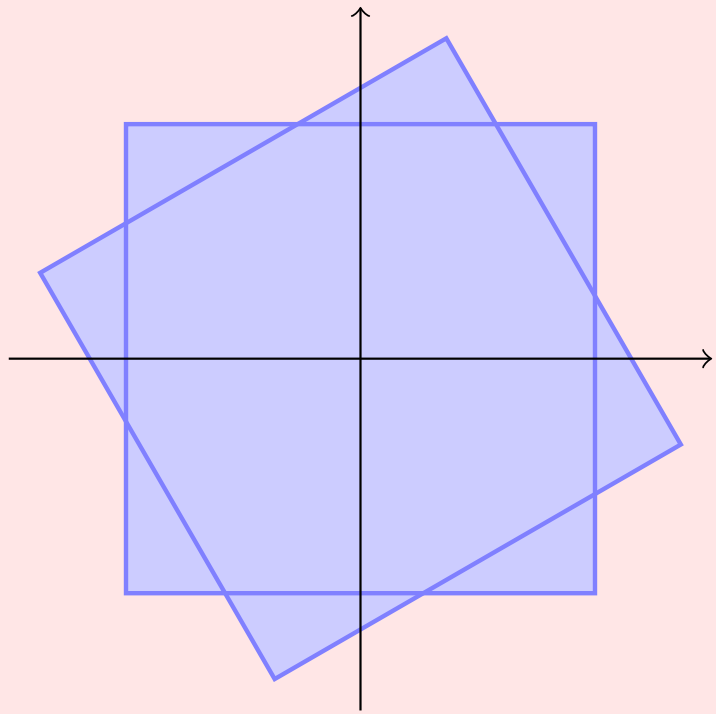


En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.

# Polyèdre, treillis

## Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.

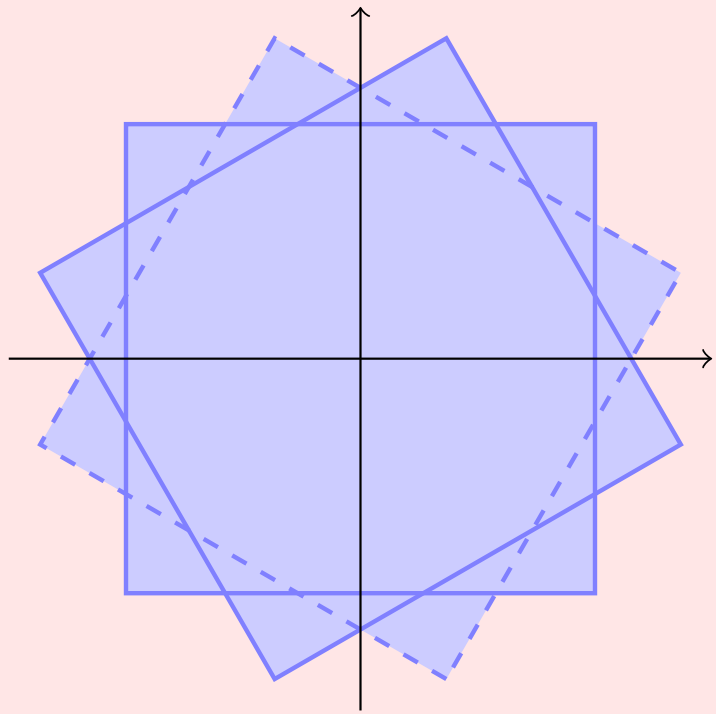


En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.

# Polyèdre, treillis

## Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.

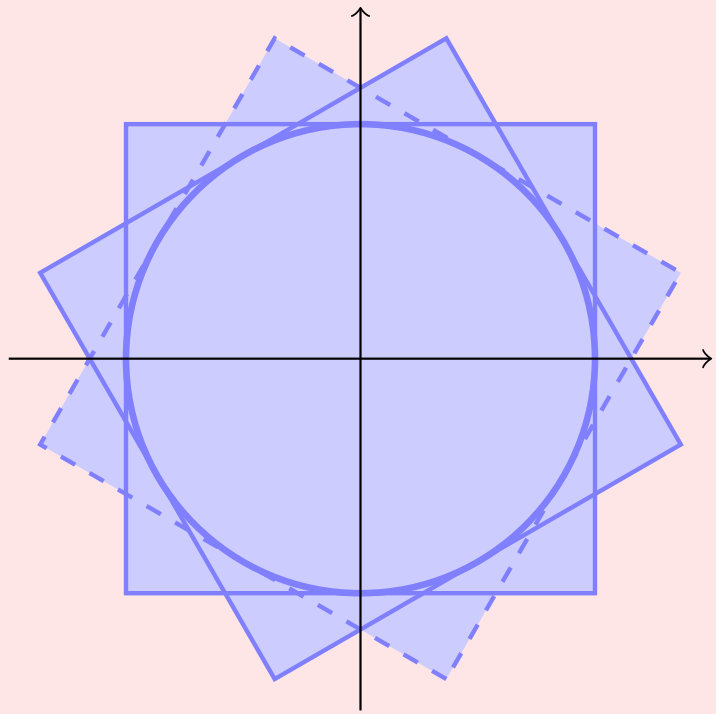


En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.

# Polyèdre, treillis

## Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.

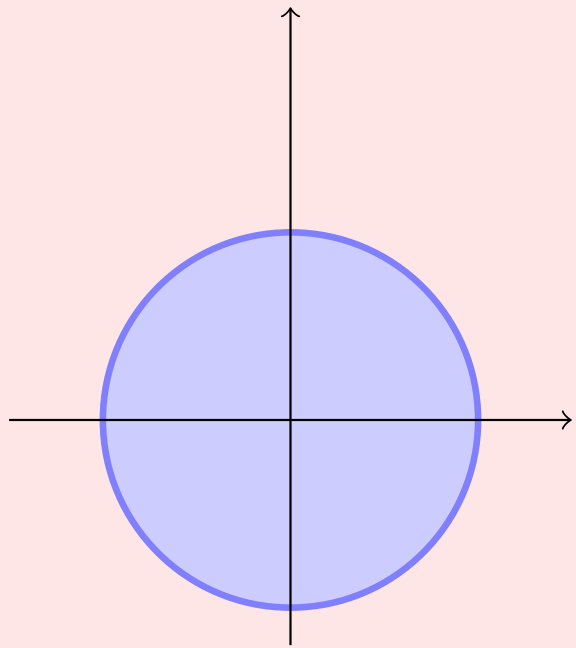


En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.

# Polyèdres, meilleure abstraction

## Remarque

De nombreux objets concrets n'ont pas de meilleure abstraction : un disque peut être approximé par un polygone régulier à $n$ côtés, un polygone régulier à $2n$ côtés sera une meilleure abstraction.

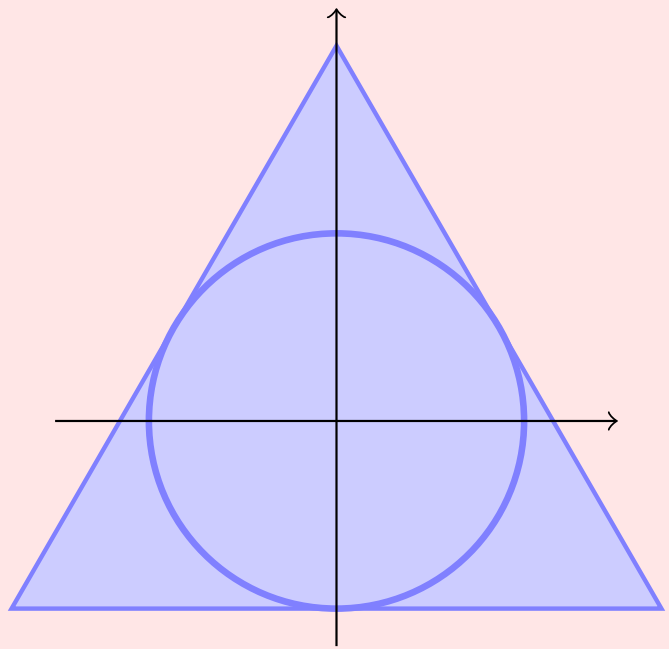


En pratique, on ne considère que des polyèdres avec un nombre fini de côtés, donc ce n'est pas gênant.

# Polyèdres, meilleure abstraction

## Remarque

De nombreux objets concrets n'ont pas de meilleure abstraction : un disque peut être approximé par un polygone régulier à $n$ côtés, un polygone régulier à $2n$ côtés sera une meilleure abstraction.

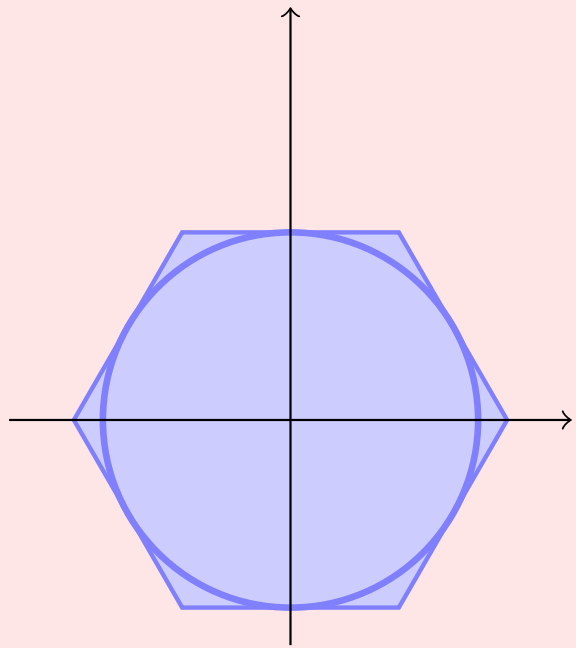


En pratique, on ne considère que des polyèdres avec un nombre fini de côtés, donc ce n'est pas gênant.

# Polyèdres, meilleure abstraction

## Remarque

De nombreux objets concrets n'ont pas de meilleure abstraction : un disque peut être approximé par un polygone régulier à $n$ côtés, un polygone régulier à $2n$ côtés sera une meilleure abstraction.



En pratique, on ne considère que des polyèdres avec un nombre fini de côtés, donc ce n'est pas gênant.

# Représentation des polyèdres

Deux représentations duales :

## Contraintes

$(M, c)$ avec $M \in \mathbb{Z}^{m \times n}$ et $c \in \mathbb{Z}^m$ :

$$\gamma(M, c) = \{v \mid Mv \geqslant c\}$$

avec $v = (v_1, \ldots, v_n)$ vecteur des variables ($v_i \in \mathbb{V}$).

# Représentation des polyèdres

Deux représentations duales :

## Contraintes

$(M, c)$ avec $M \in \mathbb{Z}^{m \times n}$ et $c \in \mathbb{Z}^m$ :

$$\gamma(M, c) = \{v \mid Mv \geqslant c\}$$

avec $v = (v_1, \ldots, v_n)$ vecteur des variables ($v_i \in \mathbb{V}$).

## Générateurs

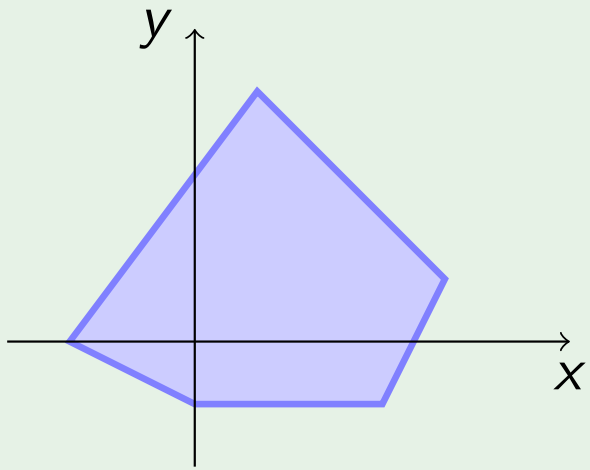
$(P, R)$ avec $P \in \mathbb{Z}^{n \times p}$ et $R \in \mathbb{Z}^{n \times r}$ :

$$\gamma(P, R) = \left\{ \left( \sum_{i=1}^{p} a_i P_{.i} \right) + \left( \sum_{i=1}^{r} b_i R_{.i} \right) \;\middle|\; \begin{array}{l} \forall i, a_i \geqslant 0, b_i \geqslant 0 \\ \sum_{i=1}^{p} a_i = 1 \end{array} \right\}$$

$P$ est nommé ensemble de *points* et $R$ ensemble de *rayons*.

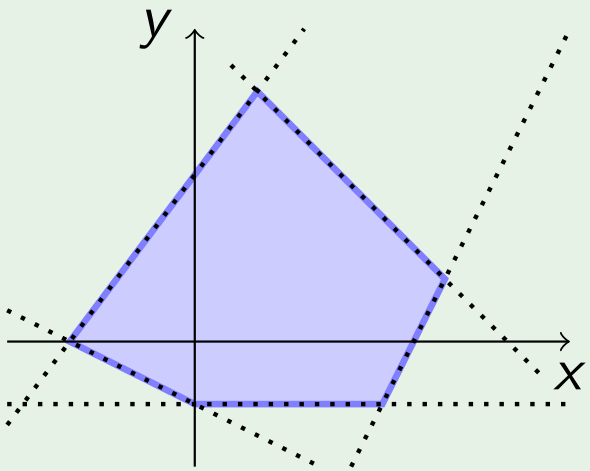
# Représentation des polyèdres, exemples

## Contraintes



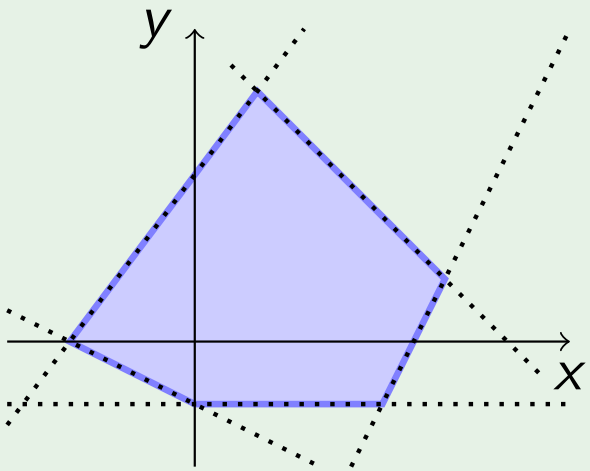
# Représentation des polyèdres, exemples

## Contraintes

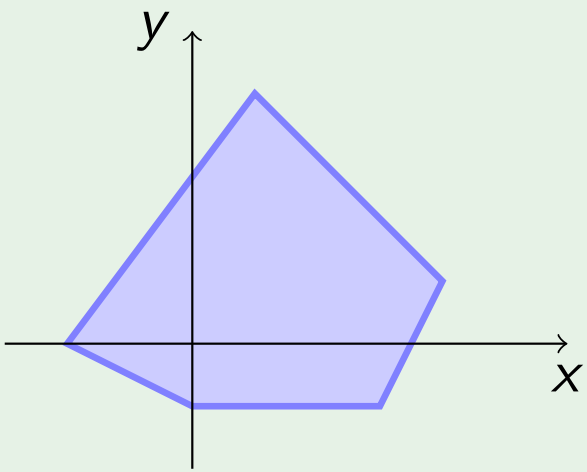


# Représentation des polyèdres, exemples

## Contraintes

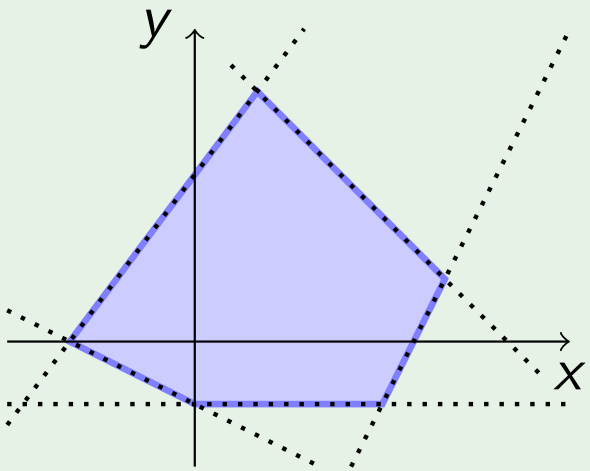



## Générateurs

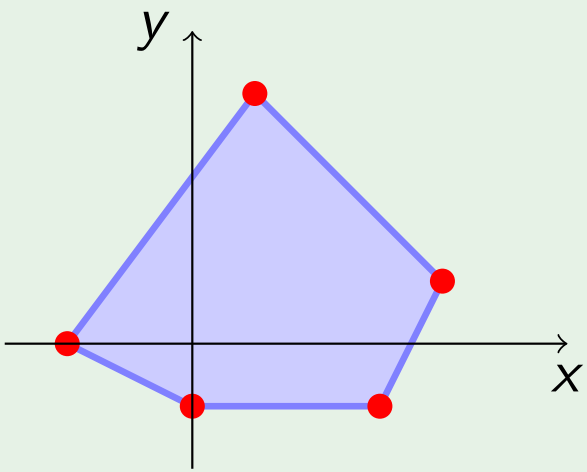


# Représentation des polyèdres, exemples

## Contraintes

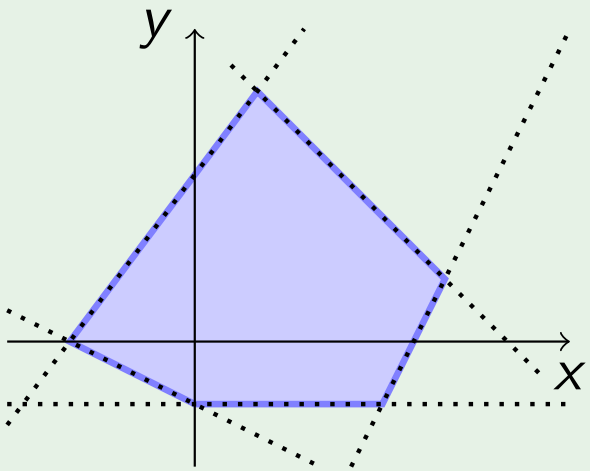


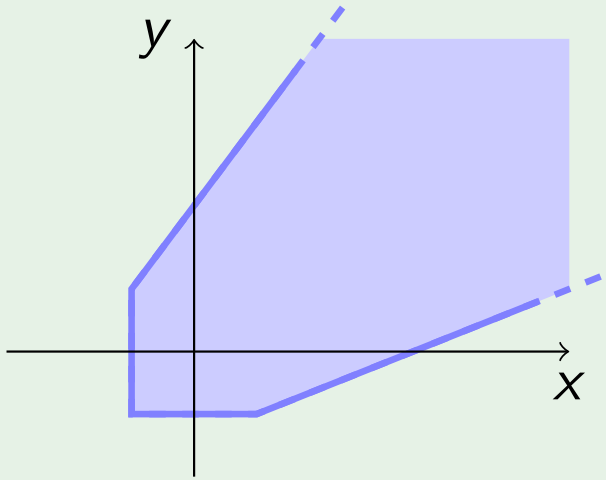

## Générateurs



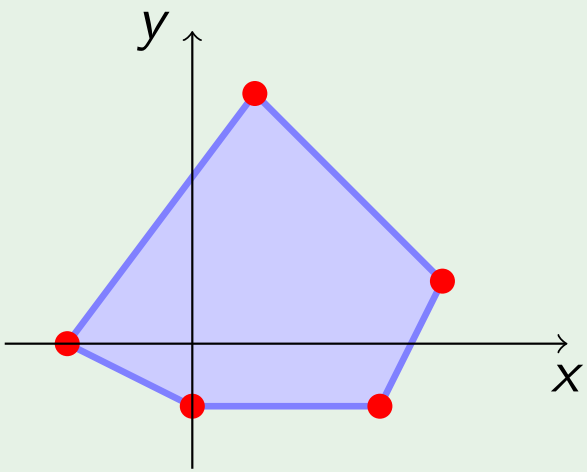
# Représentation des polyèdres, exemples

## Contraintes



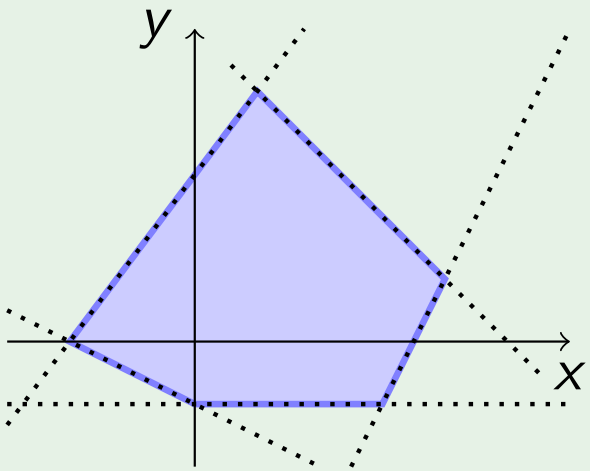



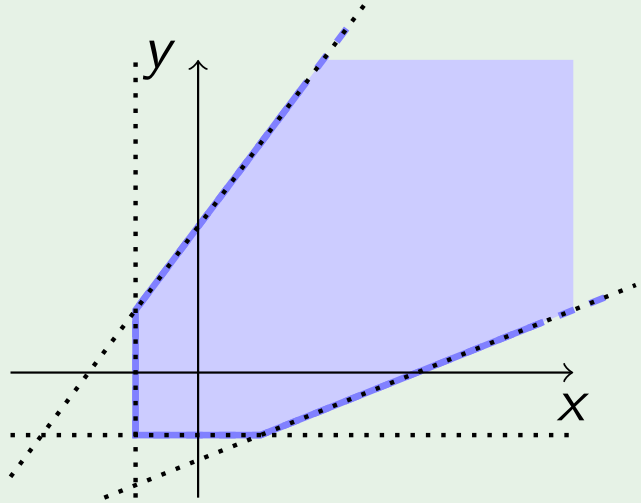

## Générateurs



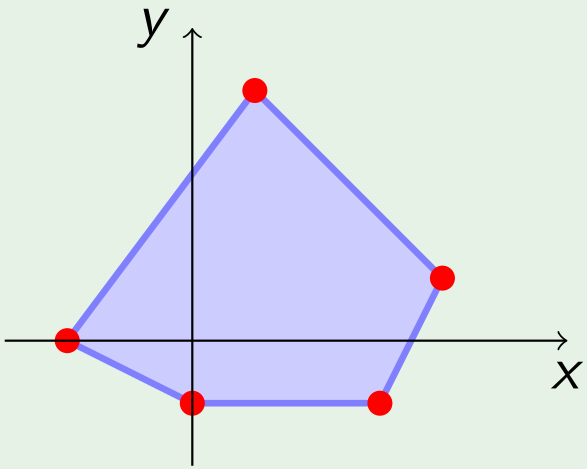
# Représentation des polyèdres, exemples

## Contraintes



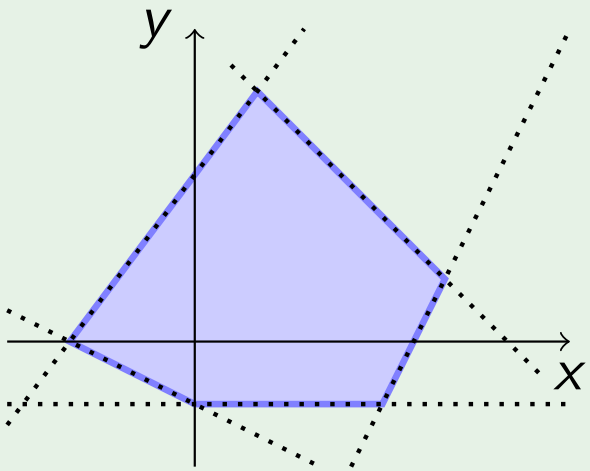



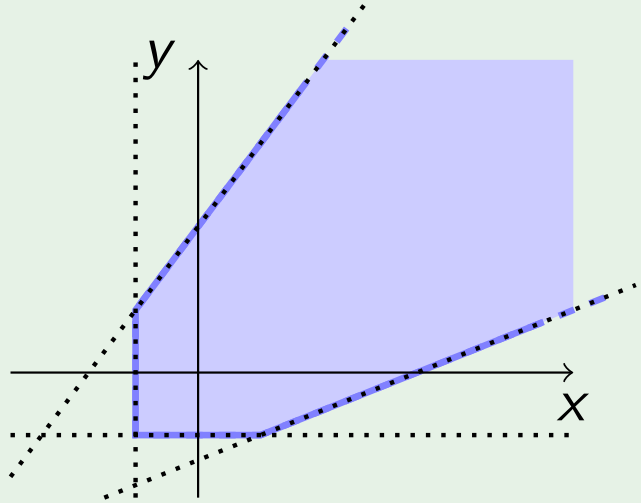

## Générateurs



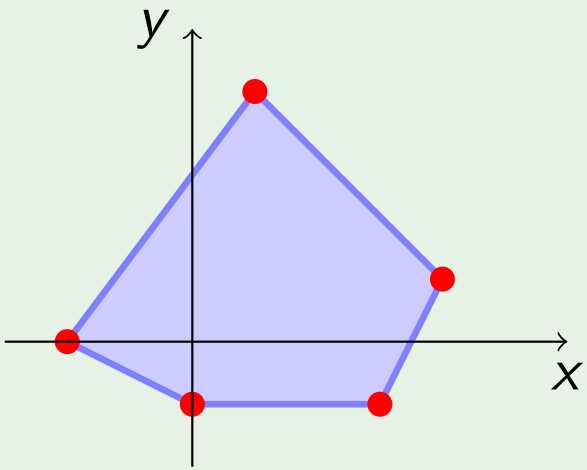
# Représentation des polyèdres, exemples

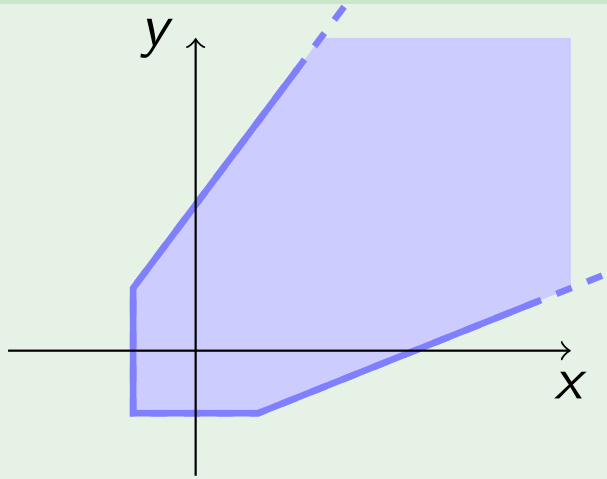
## Contraintes



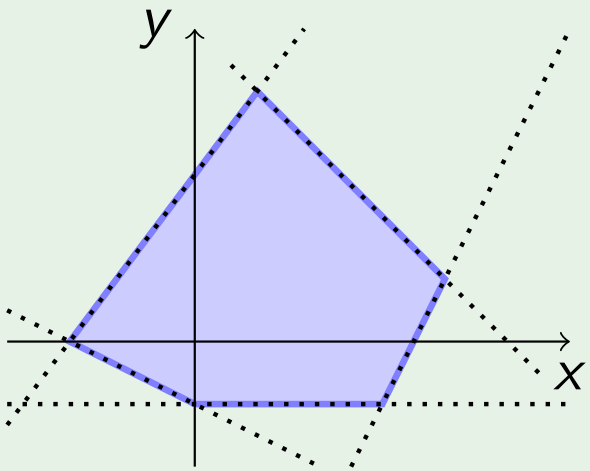

## Générateurs

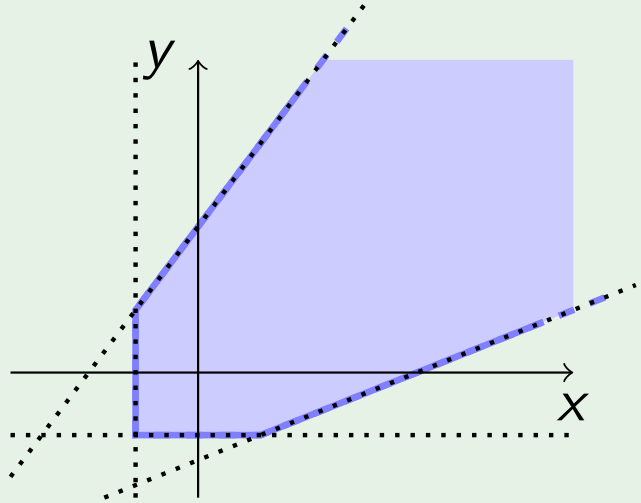


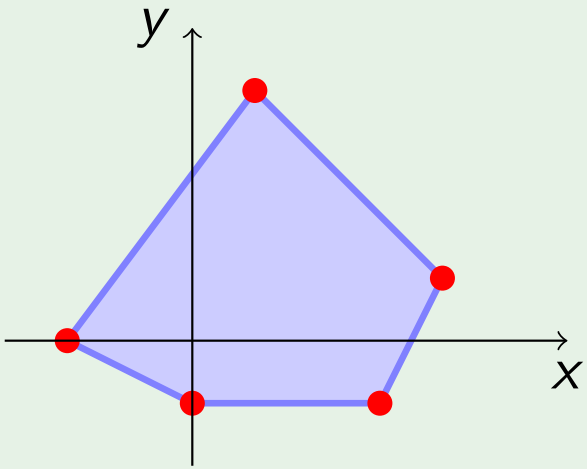
# Représentation des polyèdres, exemples

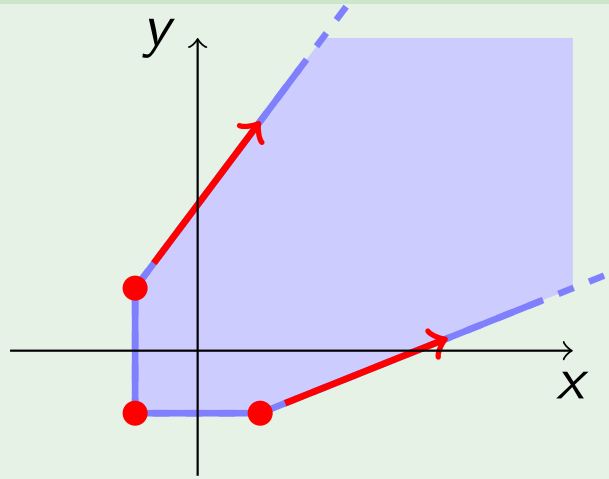
## Contraintes




## Générateurs



# Minimalité de la représentation

## Définition

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).

# Minimalité de la représentation

## Définition

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).

## Remarques

- La représentation minimale n'est pas unique.
  - contraintes

# Minimalité de la représentation

## Définition

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).

## Remarques

- La représentation minimale n'est pas unique.
  - contraintes

  - générateurs

# Minimalité de la représentation

## Définition

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).

## Remarques

- La représentation minimale n'est pas unique.
  - contraintes

  - générateurs

- Il est intéressant de garder une représentation minimale pour minimiser la complexité spatiale et temporelle.

# Remarques sur la dualité

## Remarques

- Les opérations sont souvent plus faciles sur une des représentations que sur l'autre.
- On a un algorithme (Chernikova) pour passer d'une représentation à l'autre.
- Complexité au pire cas exponentielle en $n$ (l'hypercube de dimension $n$ a $2n$ faces et $2^n$ sommets).

# Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$

# Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$

# Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$
- $x^\sharp \sqcap^\sharp y^\sharp$ : union des ensembles de contraintes

# Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$
- $x^\sharp \sqcap^\sharp y^\sharp$ : union des ensembles de contraintes
- $x^\sharp \sqcup^\sharp y^\sharp$ : union des ensembles de générateurs

# Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$
- $x^\sharp \sqcap^\sharp y^\sharp$ : union des ensembles de contraintes
- $x^\sharp \sqcup^\sharp y^\sharp$ : union des ensembles de générateurs

- Gardes : on ajoute des contraintes :

$$\left[\!\left[ \sum_i a_i v_i + b > 0 \right]\!\right]^\sharp_{\mathrm{C}} (M, c) = \left( \begin{pmatrix} M \\ a_1 \dots a_n \end{pmatrix}, \begin{pmatrix} c \\ 1 - b \end{pmatrix} \right)$$

# Opérations abstraites, affectation

On applique simplement l'affectation aux générateurs :

$$\left[\!\left[ v_i = \sum_i a_i v_i + b \right]\!\right]^\sharp_{\mathrm{C}} (P, R) = (AP + B, AR)$$

avec

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ a_1 & \cdots & a_i & \cdots & a_n \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 0 \\ \vdots \\ b \\ \vdots \\ 0 \end{pmatrix}$$

# Opérations abstraites, affectation

On applique simplement l'affectation aux générateurs :

$$\left[\!\left[ v_i = \sum_i a_i v_i + b \right]\!\right]^\sharp_{\mathrm{C}} (P, R) = (AP + B, AR)$$

avec

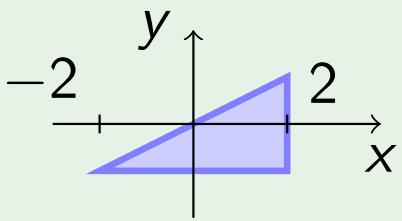
$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ a_1 & \cdots & a_i & \cdots & a_n \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 0 \\ \vdots \\ b \\ \vdots \\ 0 \end{pmatrix}$$

## Remarques

- Malgré l'absence de correspondance de Galois, toutes ces opérations sont optimales (et même exactes, sauf $\sqcup^\sharp$).
- Dans le cas non linéaire, il faudrait abstraire par du linéaire...

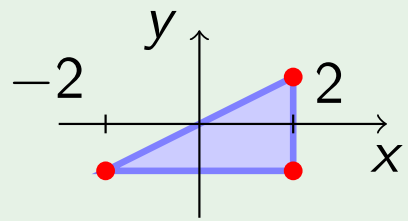
# Opérations abstraites, affectation, exemples

## Exemple ($x = x - y - 1$)



# Opérations abstraites, affectation, exemples

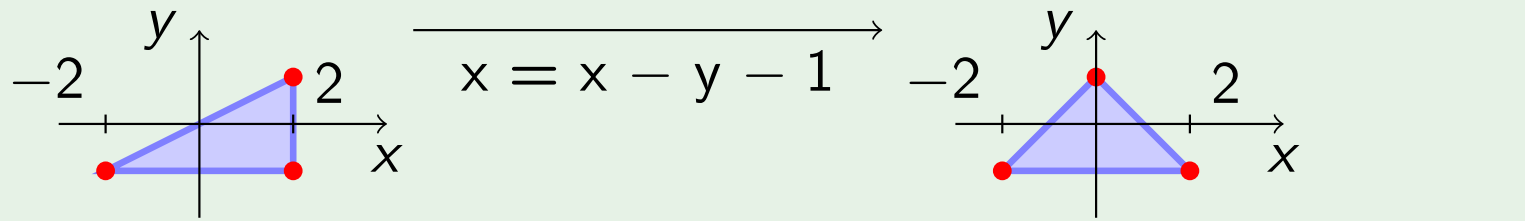
## Exemple ($x = x - y - 1$)




$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$$

# Opérations abstraites, affectation, exemples

## Exemple ($x = x - y - 1$)

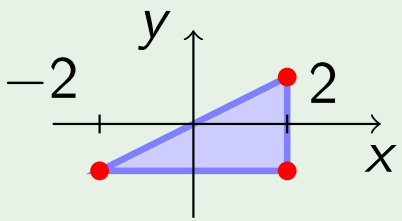



$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$$

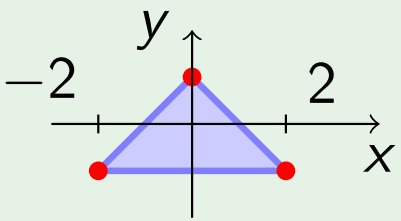
$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$$

# Opérations abstraites, affectation, exemples

## Exemple (x = x − y − 1)



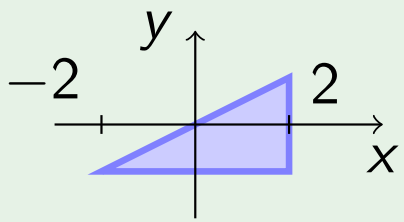

$$x = x - y - 1$$




$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$$

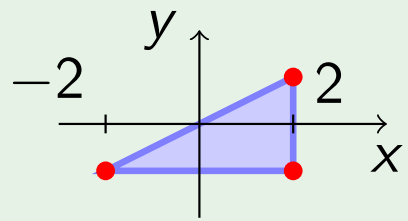
$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$$

## Exemple (x = 2y)

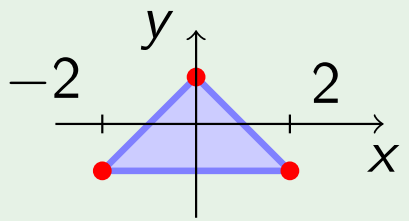


# Opérations abstraites, affectation, exemples

## Exemple (x = x − y − 1)



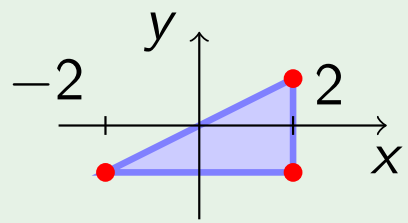

$$x = x - y - 1$$




$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$$

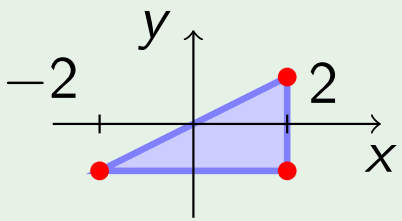
## Exemple (x = 2y)



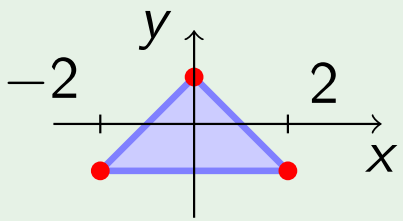

$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$$

# Opérations abstraites, affectation, exemples

## Exemple (x = x − y − 1)

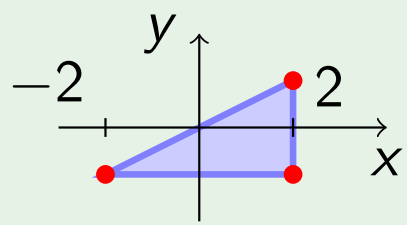



$$\xrightarrow{\quad x = x - y - 1 \quad}$$

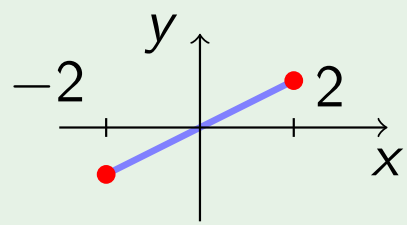



$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$$

## Exemple (x = 2y)



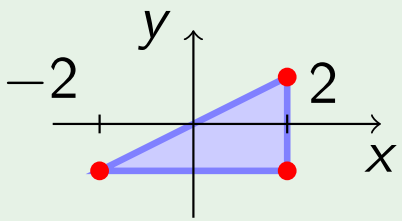

$$\xrightarrow{\quad x = 2y \quad}$$



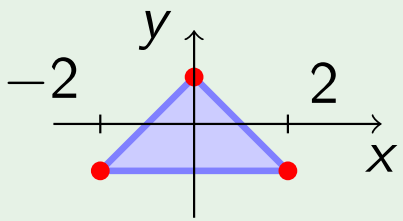

$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & -2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$$

# Opérations abstraites, affectation, exemples

## Exemple (x = x − y − 1)

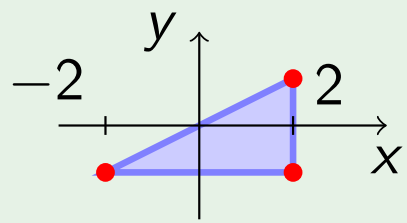



$$\xrightarrow{x = x - y - 1}$$

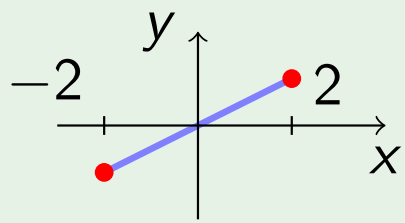



$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$$

## Exemple (x = 2y)




$$\xrightarrow{x = 2y}$$




$$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & -2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$$

## Exercice 1

Définir l'opérateur abstrait d'affectation sur les contraintes.

# Élargissement

On a des chaînes croissantes infinies
donc il nous faut un élargissement (widening).

# Élargissement

On a des chaînes croissantes infinies
donc il nous faut un élargissement (widening).

## Idée

Toujours la même : ne conserver que les contraintes stables.

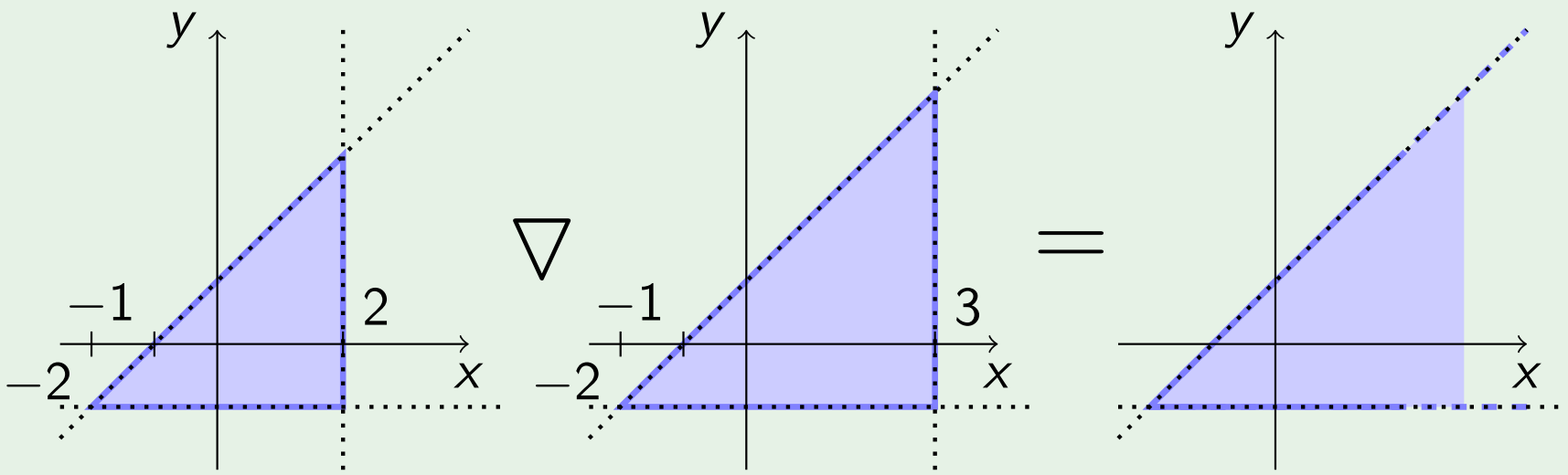
# Élargissement

On a des chaînes croissantes infinies
donc il nous faut un élargissement (widening).

## Idée

Toujours la même : ne conserver que les contraintes stables.

## Exemple



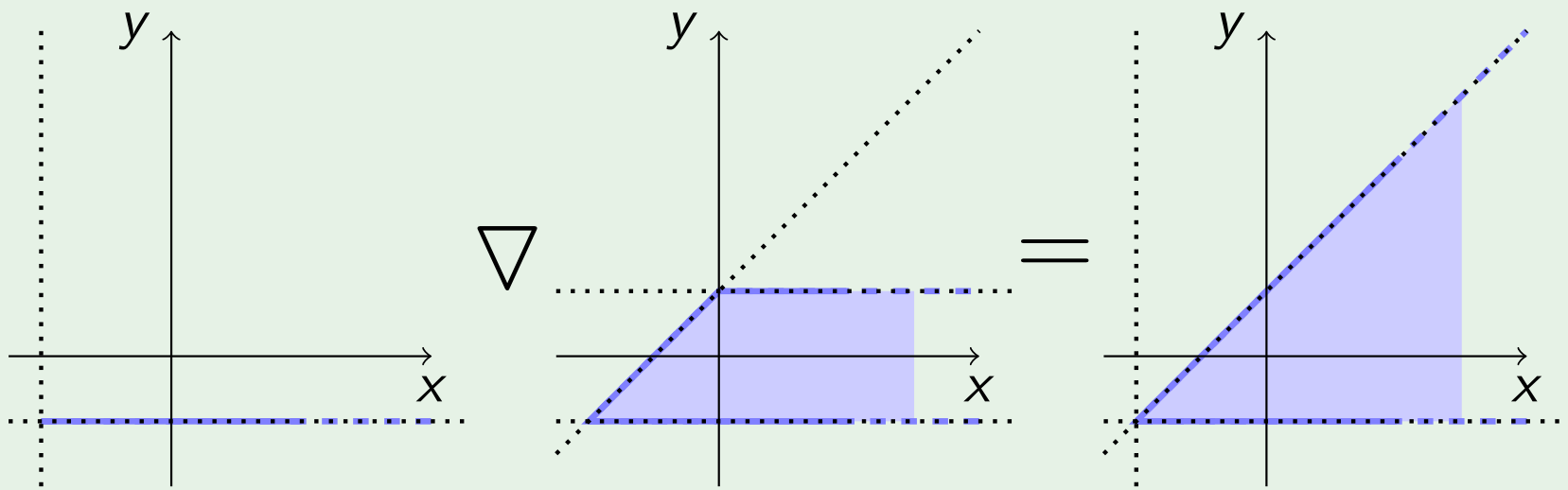
# Élargissement (suite et fin)

Plus formellement :

## Définition

Pour $x^\sharp$ et $y^\sharp$ sous forme d'ensemble de contraintes minimaux,
$x^\sharp \triangledown y^\sharp =$
$$\left\{ c \in x^\sharp \;\middle|\; y^\sharp \in \{ c \} \right\} \cup \left\{ c \in y^\sharp \;\middle|\; \exists c' \in x^\sharp, x^\sharp =^\sharp \left( x^\sharp \setminus c' \right) \cup \{ c \} \right\}$$

## Exemple



# Exemple

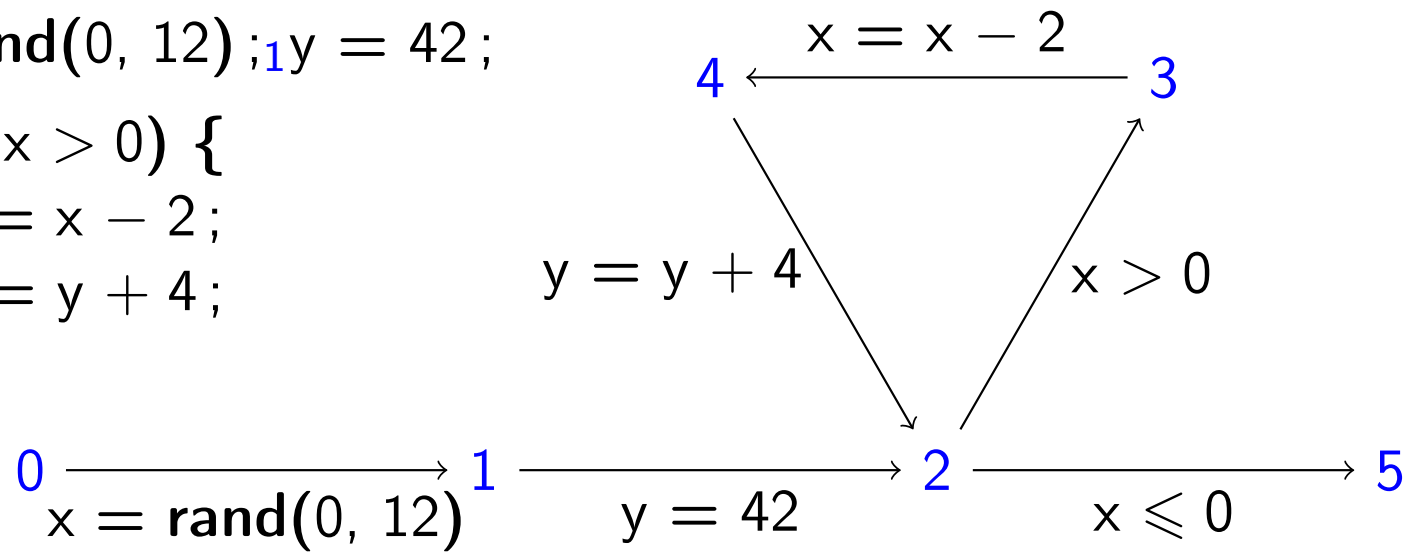
$_0$x = **rand**(0, 12) ;$_1$y = 42 ;

**while** $_2$(x > 0) **{**

    $_3$x = x − 2 ;

    $_4$y = y + 4 ;

**}**$_5$



# Exemple

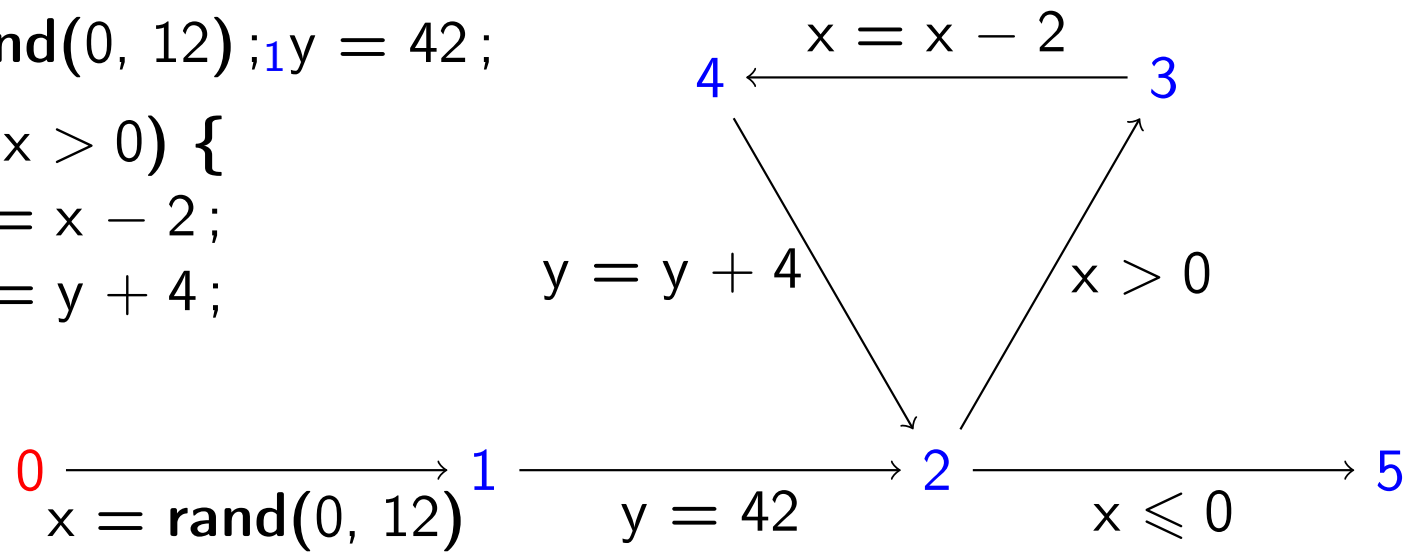
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

**}**$_5$




$\top$

0

# Exemple

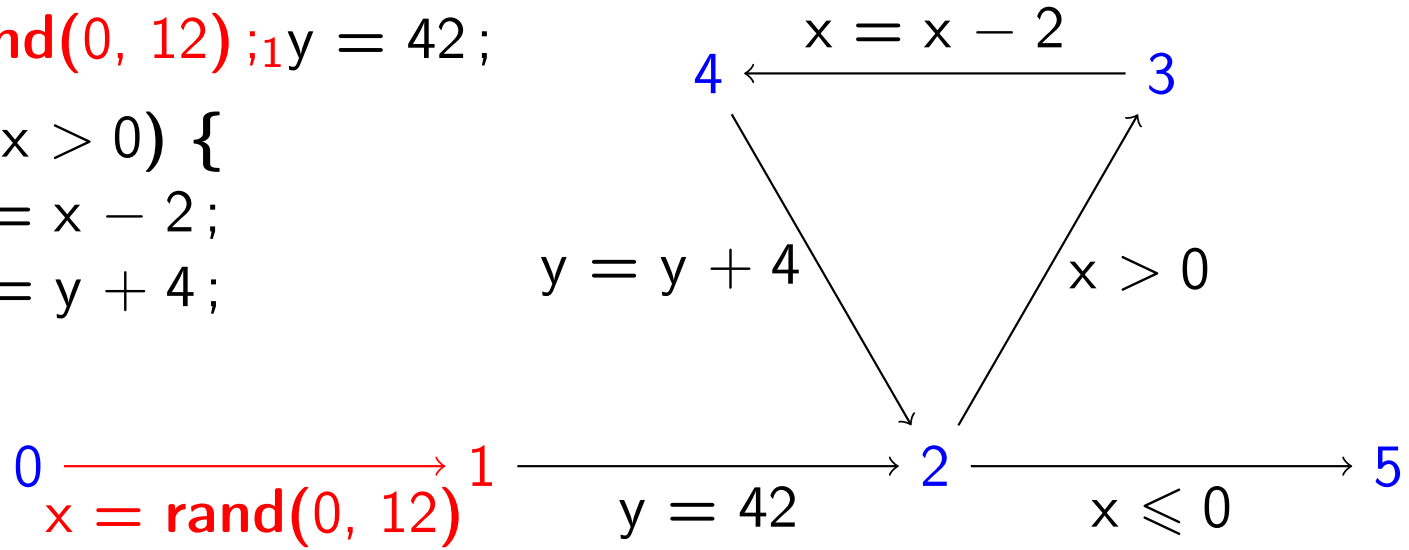
$_0x = \mathbf{rand}(0, 12)\,;_1 y = 42\,;$

**while** $_2(x > 0)$ **{**

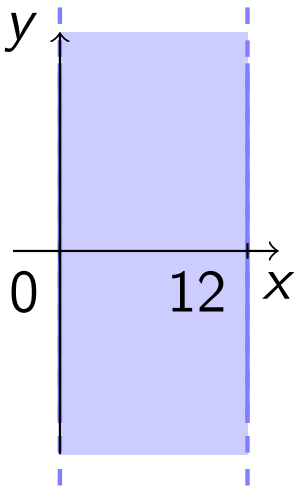
$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

**}**$_5$




$\top$




0 1

# Exemple

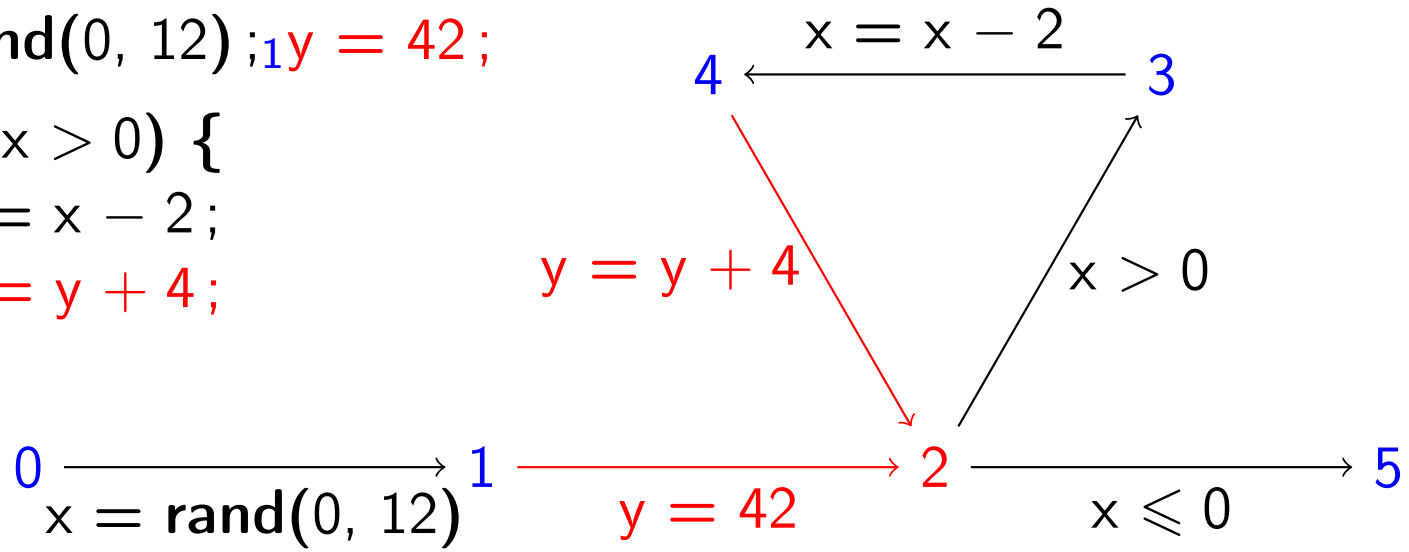
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

**while** $_2(x > 0)$ **{**

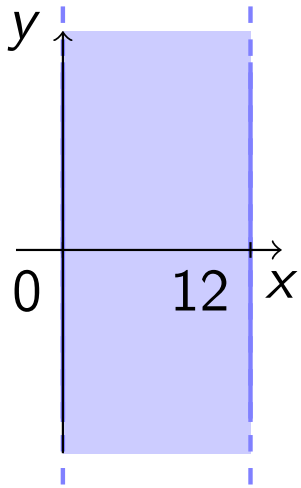
$\quad {}_3x = x - 2\,;$

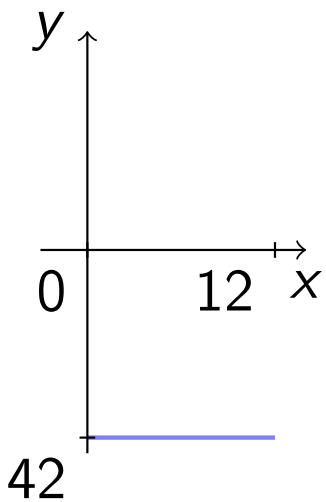
$\quad {}_4y = y + 4\,;$

$\}_5$




$\top$




0 1 2

# Exemple

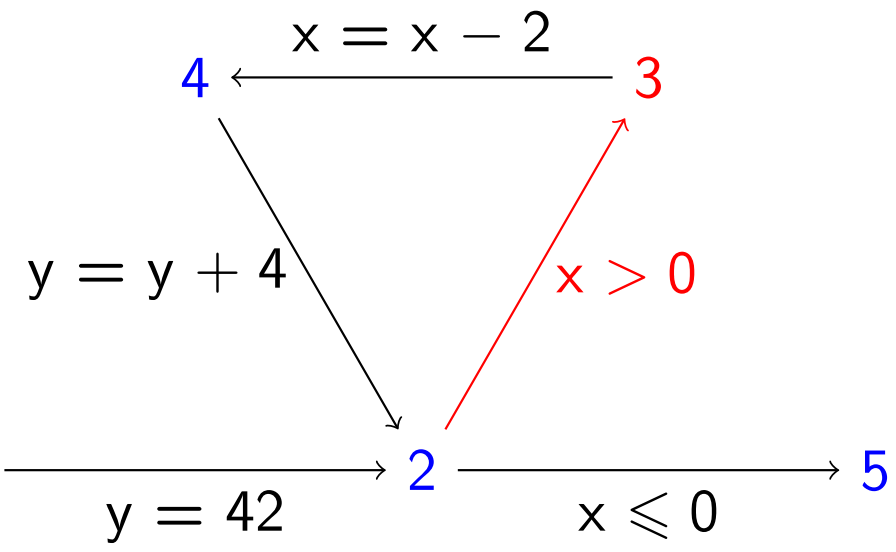
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

**while** $_2(x > 0)$ **{**

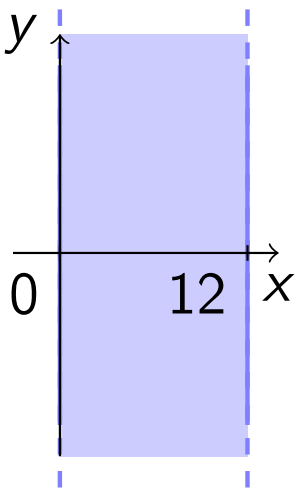
$\quad {}_3x = x - 2\,;$

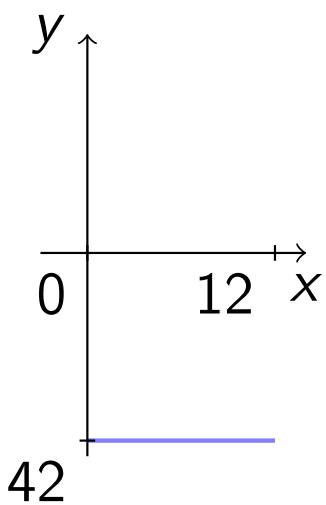
$\quad {}_4y = y + 4\,;$

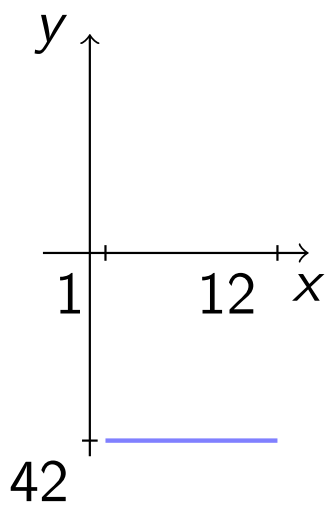
$\}_5$




$\top$




0 1 2 3

# Exemple

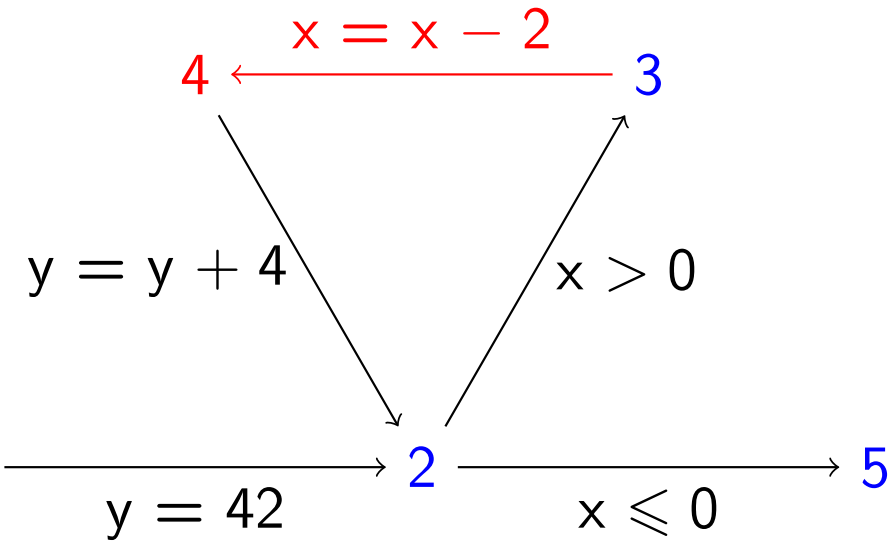
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

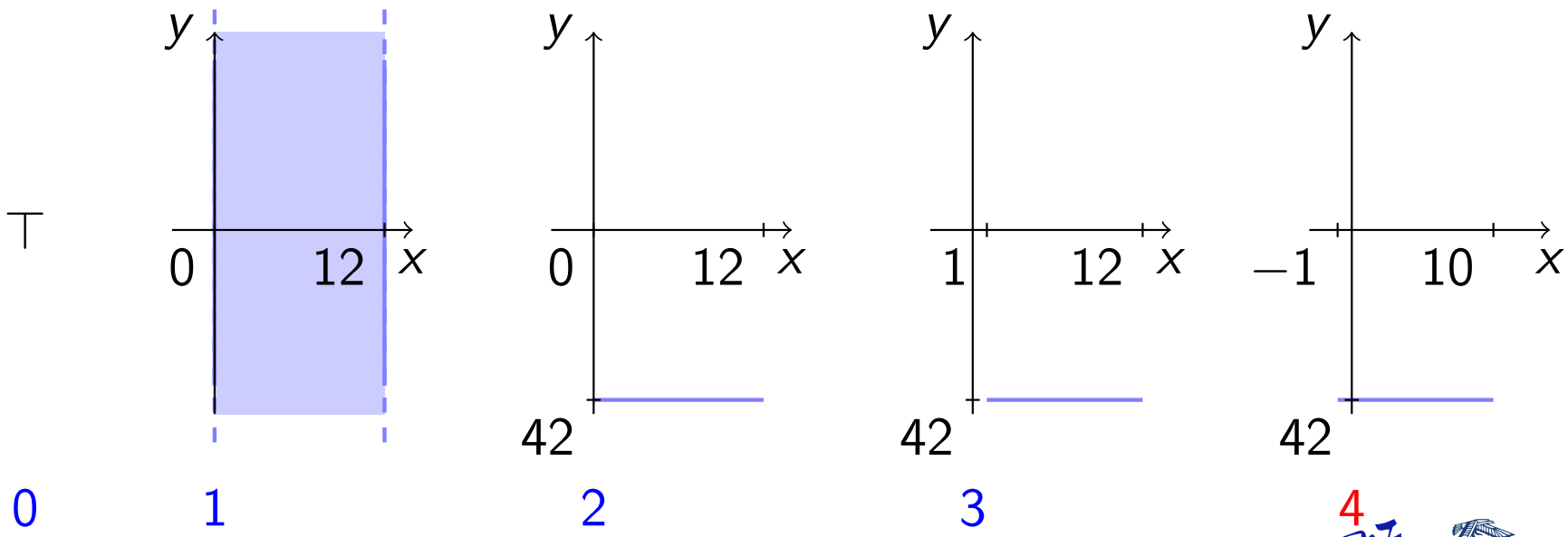
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

$\}_5$



# Exemple (suite)

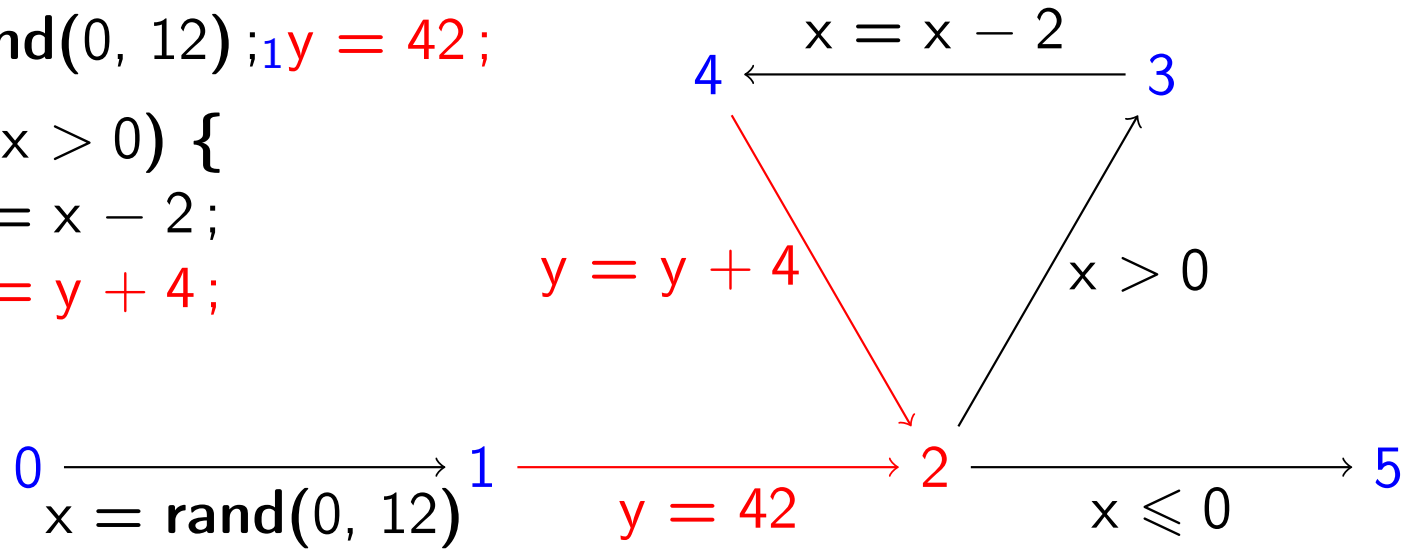
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

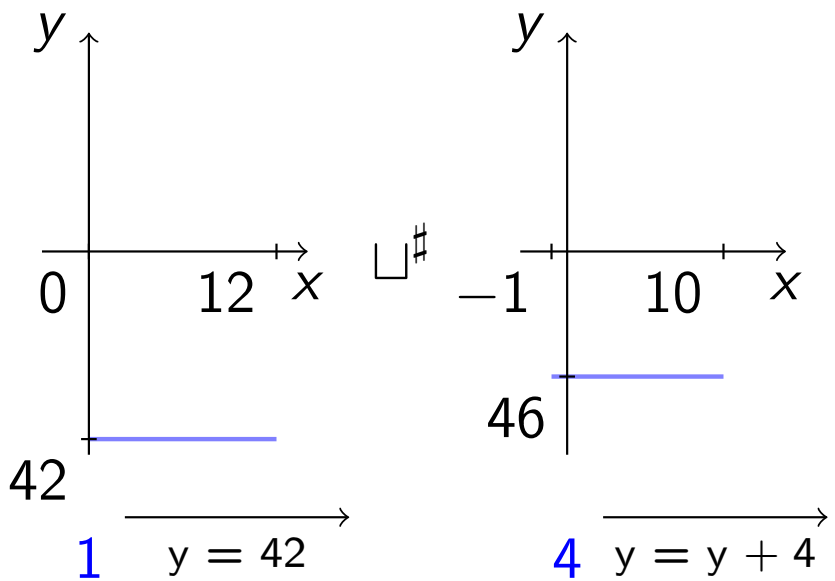
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

$\}_5$



# Exemple (suite)

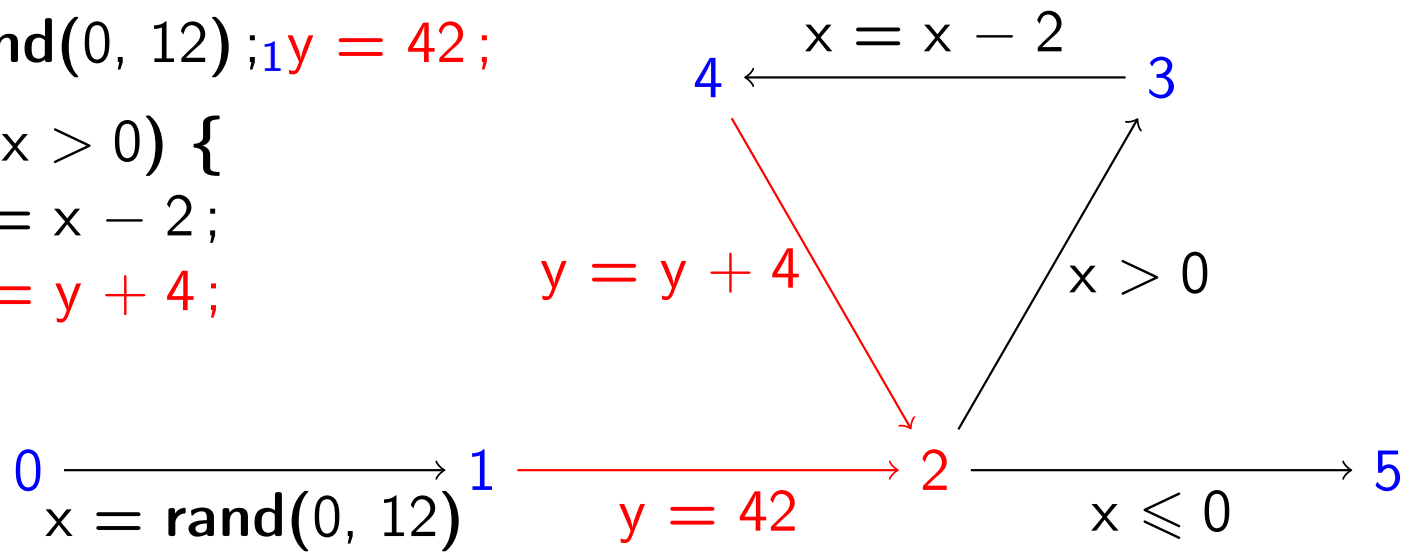
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

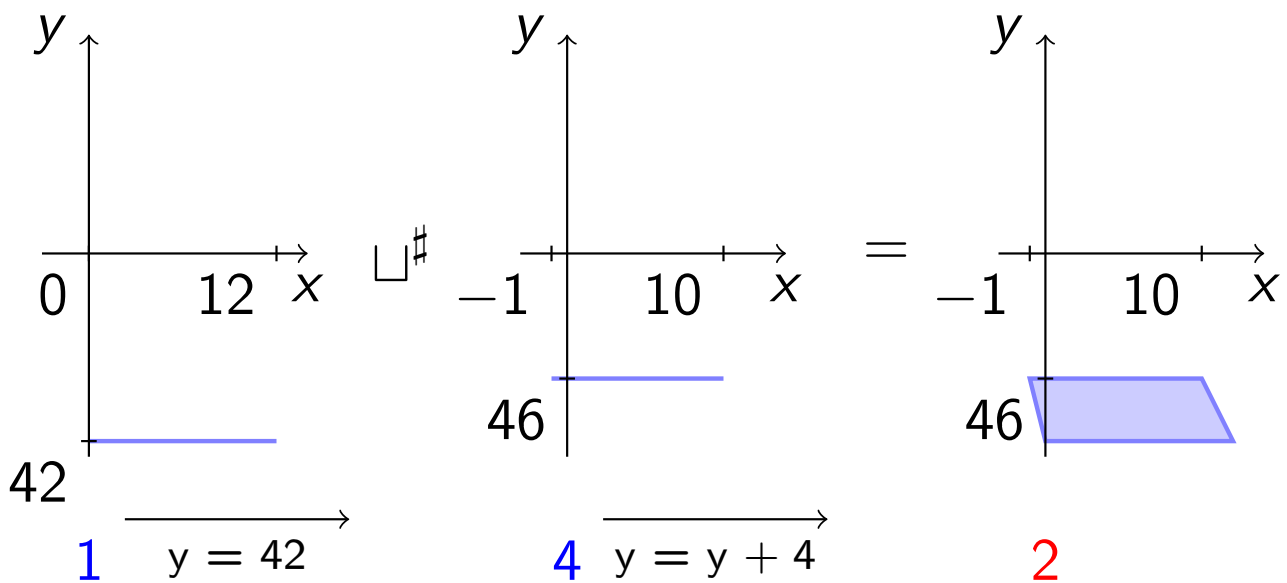
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

$\}_5$



# Exemple (suite)

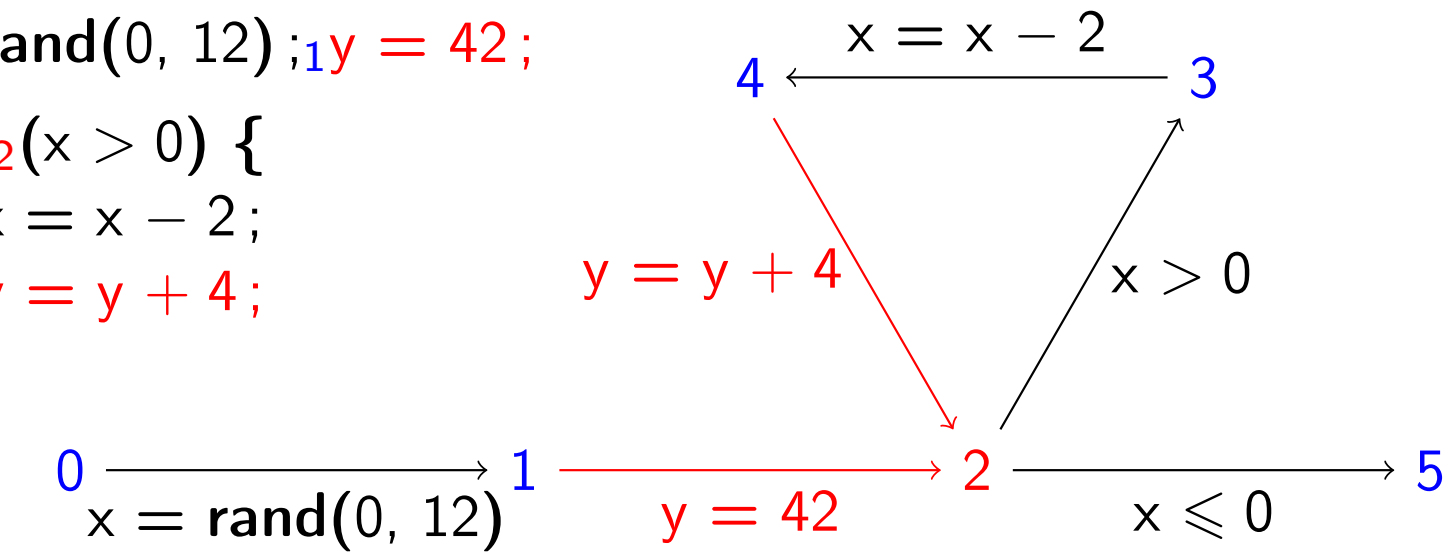
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

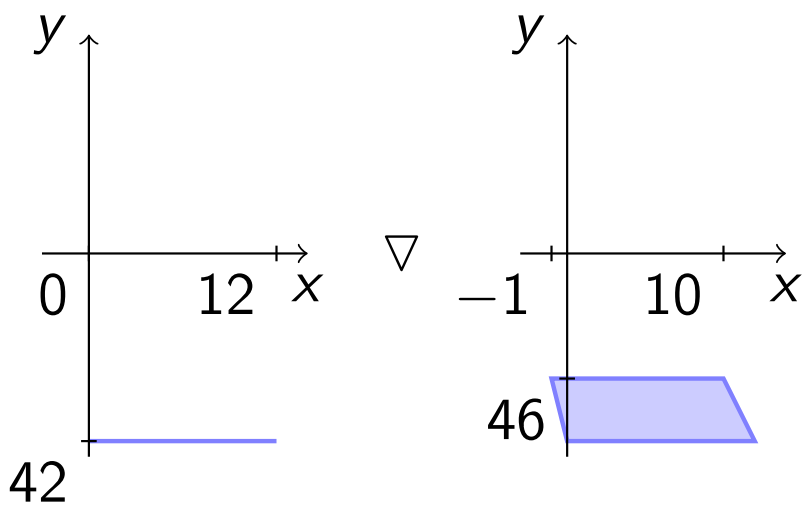
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

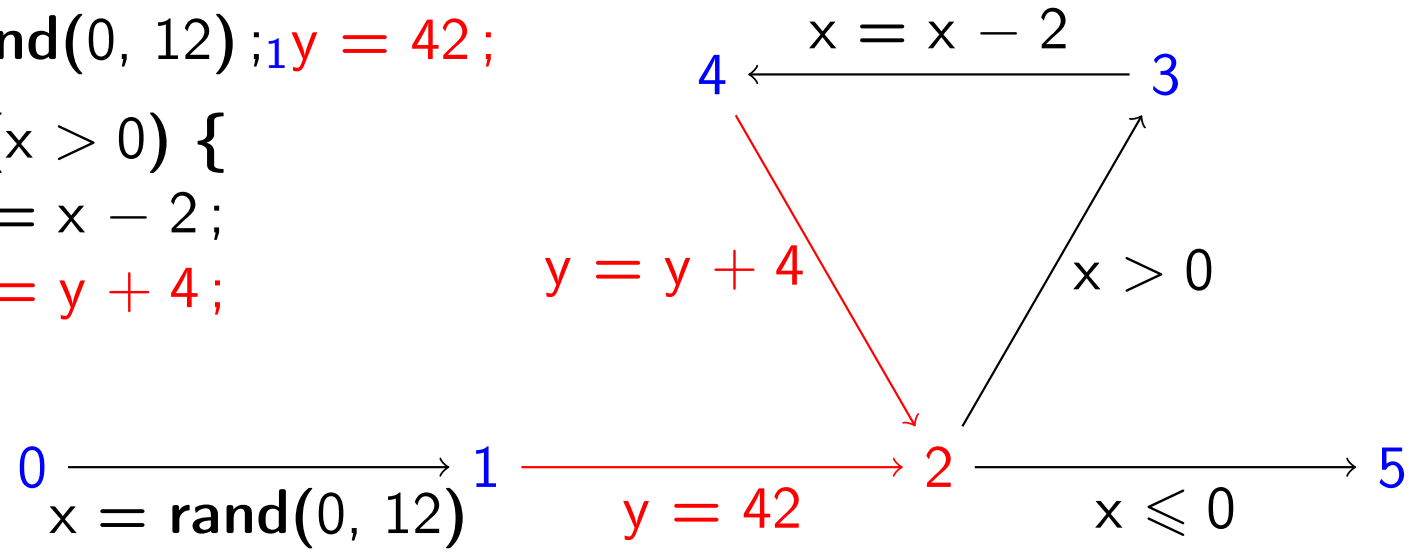
$\}_5$

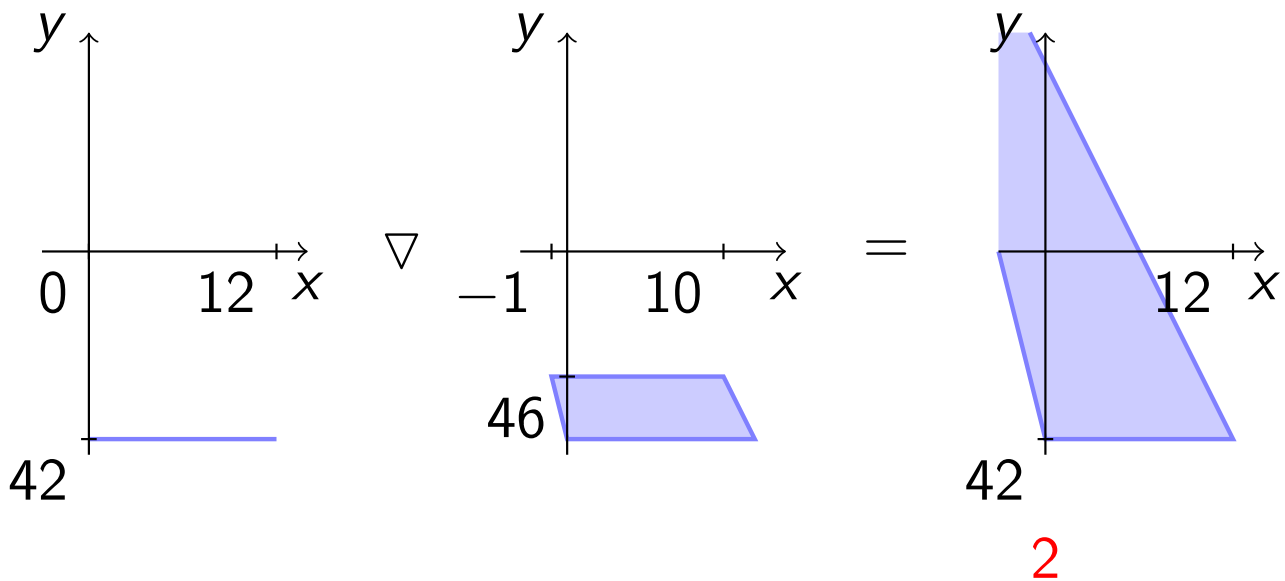


# Exemple (suite)

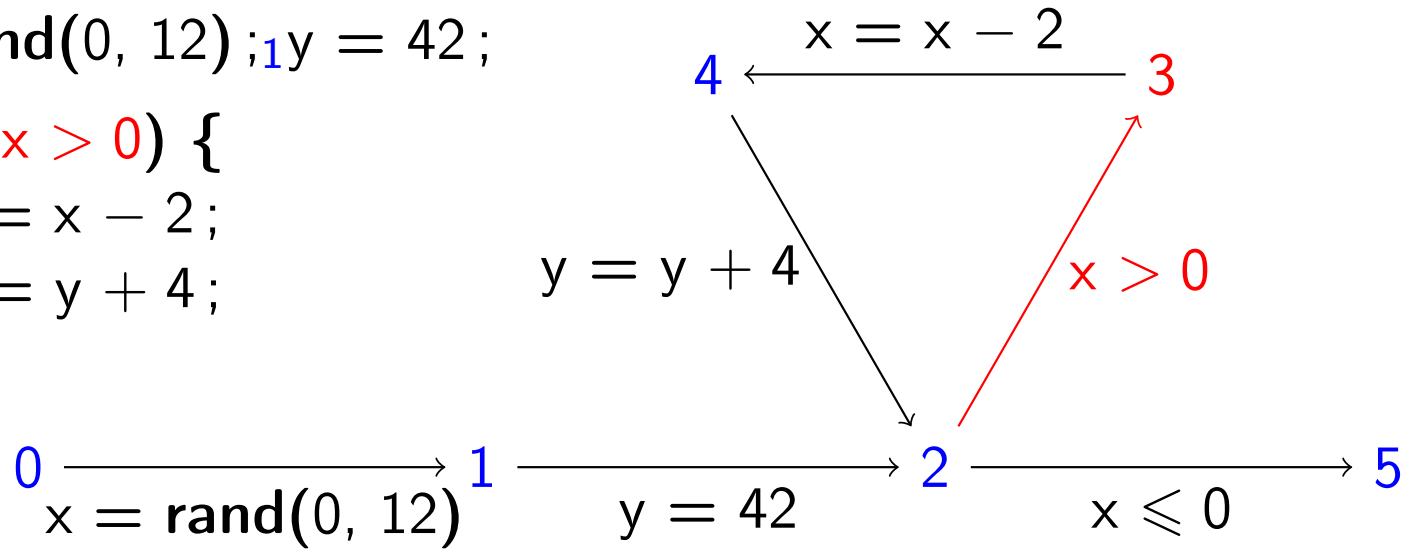
$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) {
    $_3$x = x − 2 ;
    $_4$y = y + 4 ;
}$_5$

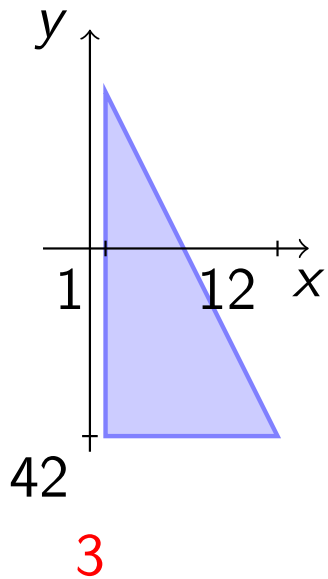


# Exemple (suite)

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) {
    $_3$x = x − 2 ;
    $_4$y = y + 4 ;
}$_5$



# Exemple (suite)

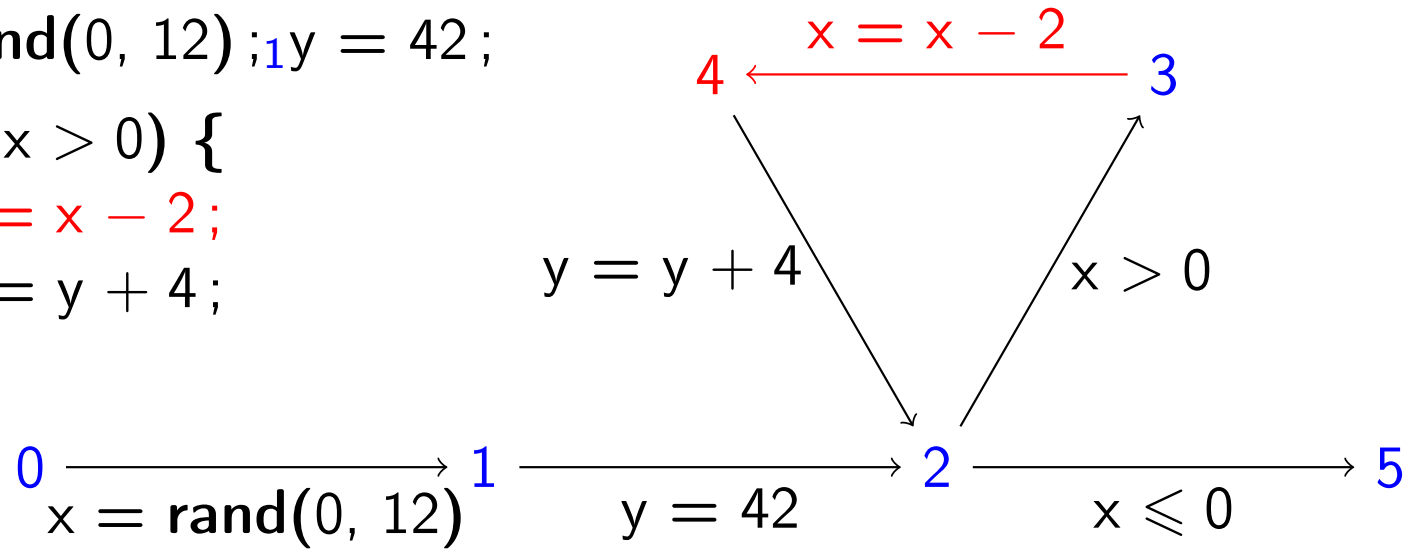
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

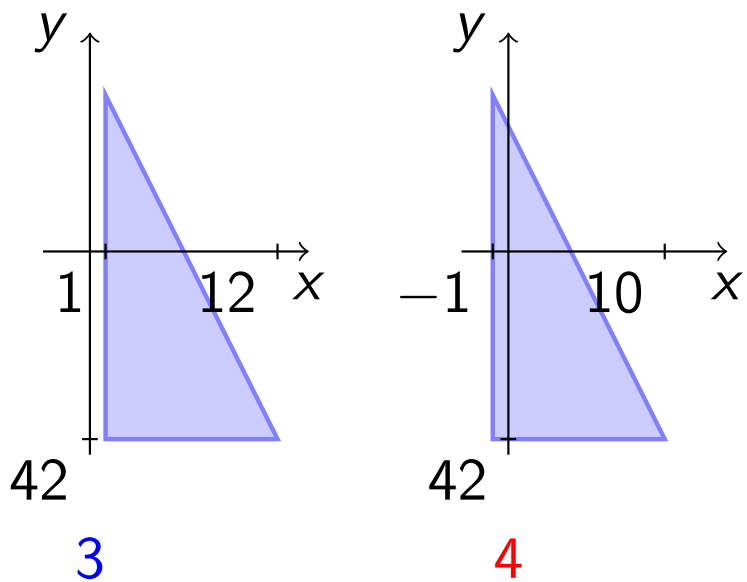
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

$\}_5$



# Exemple (suite)

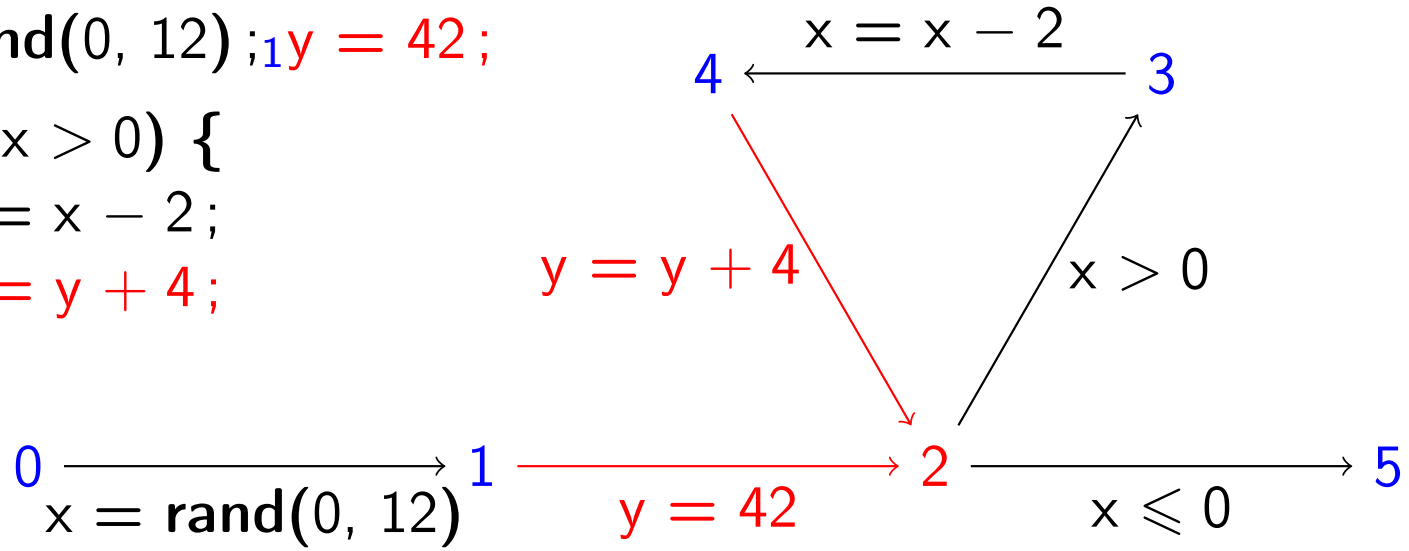
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

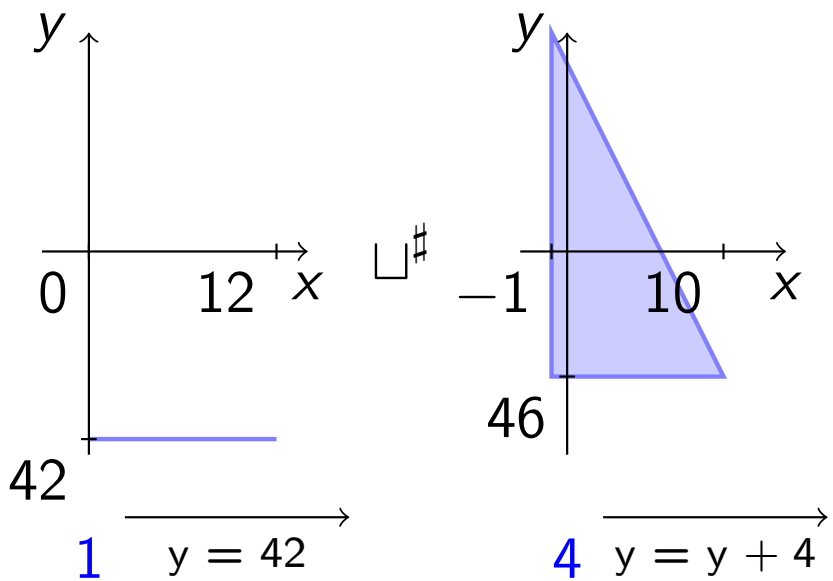
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

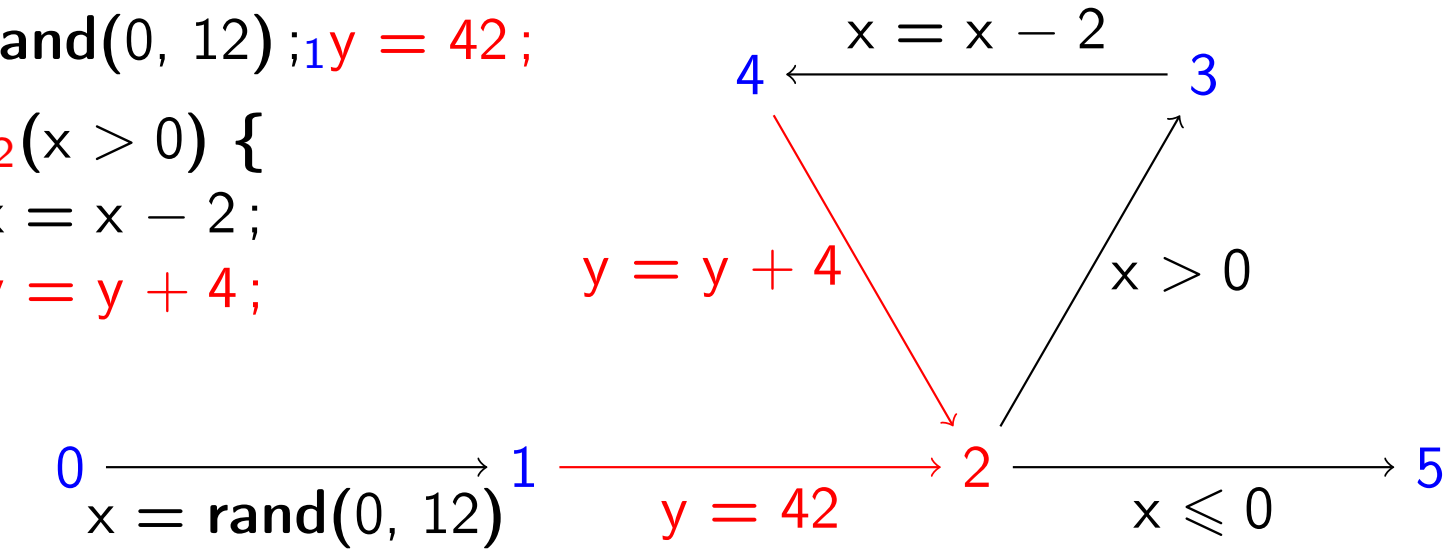
$\}_5$

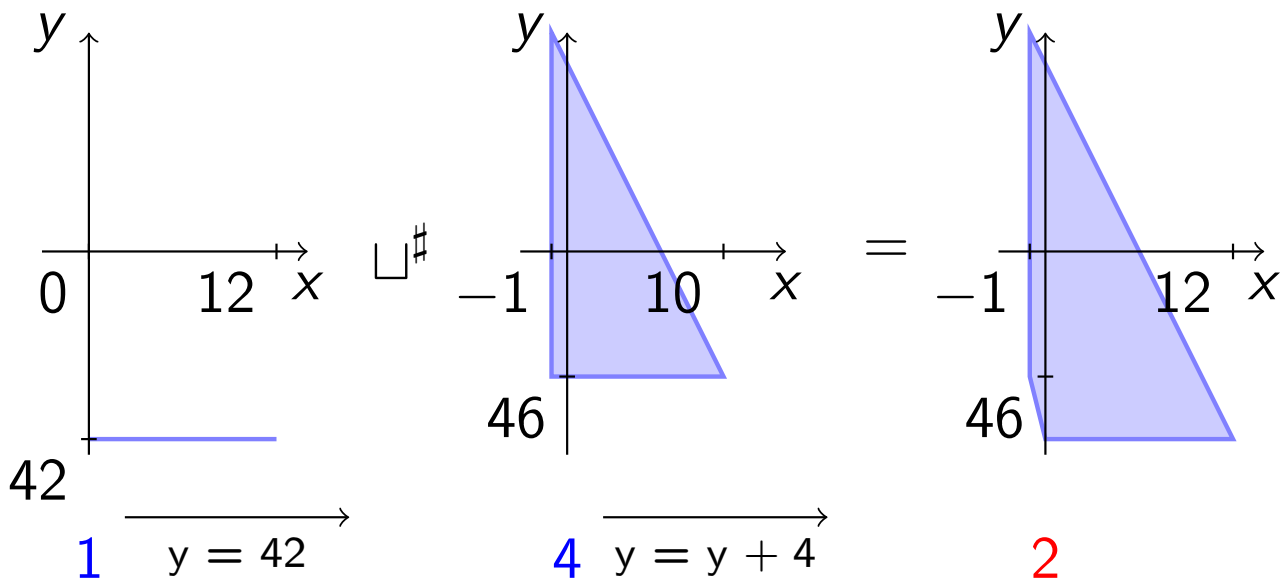


# Exemple (suite)

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) {
    $_3$x = x − 2 ;
    $_4$y = y + 4 ;
}$_5$



# Exemple (suite)

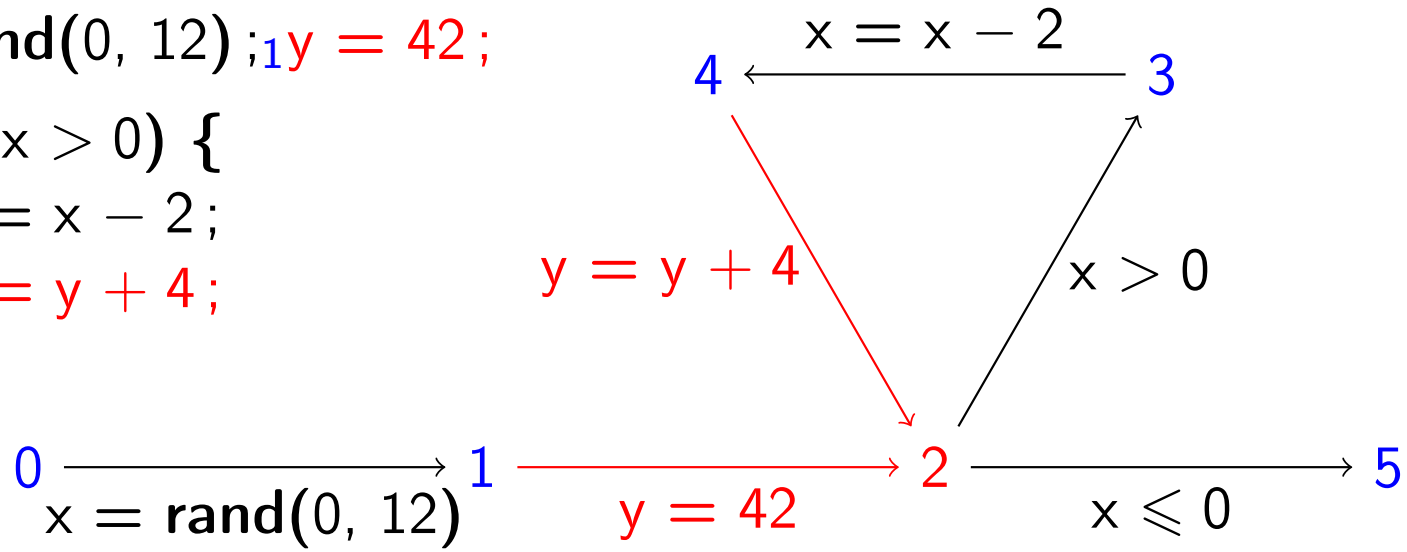
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

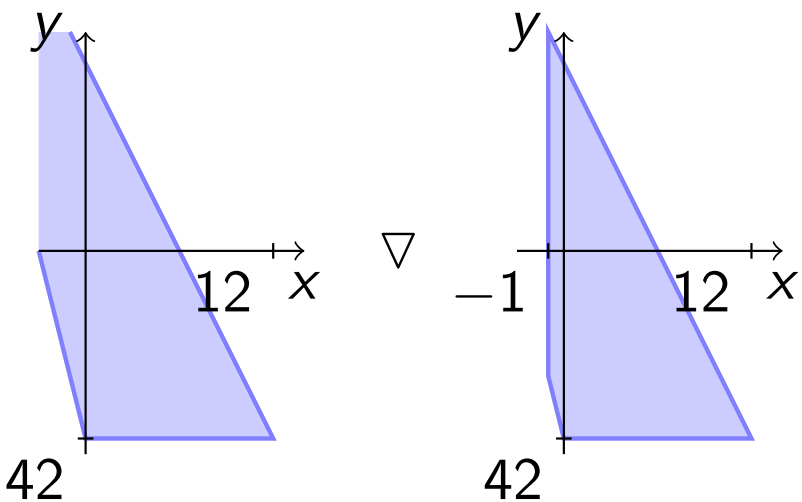
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

**}**$_5$



# Exemple (suite)

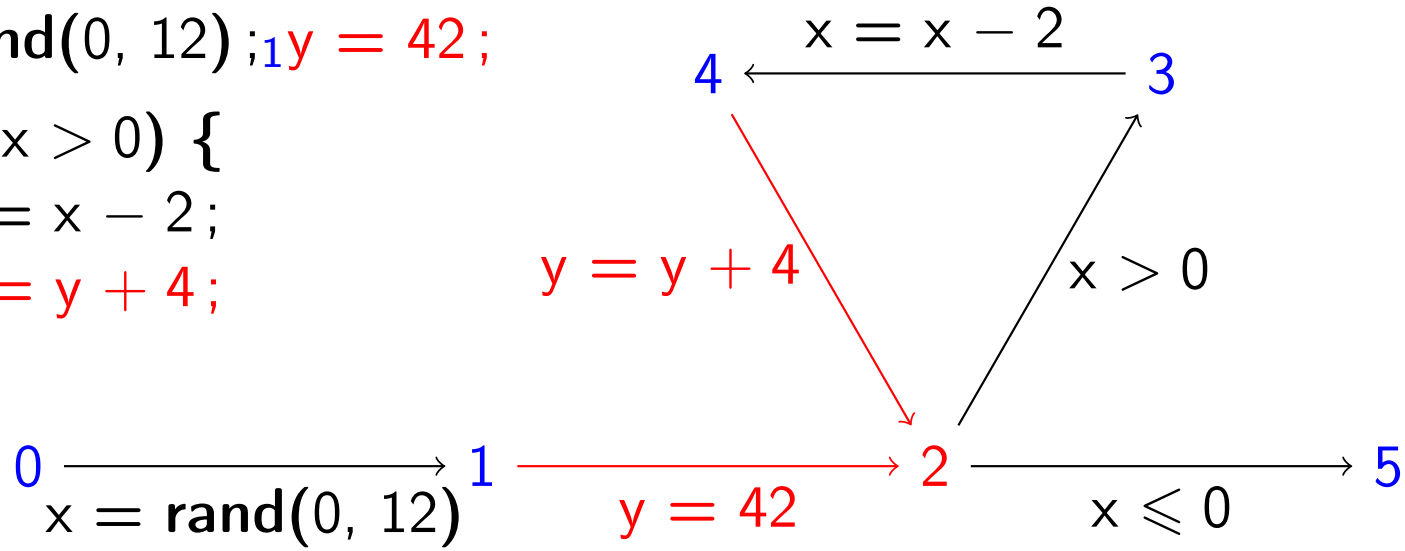
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

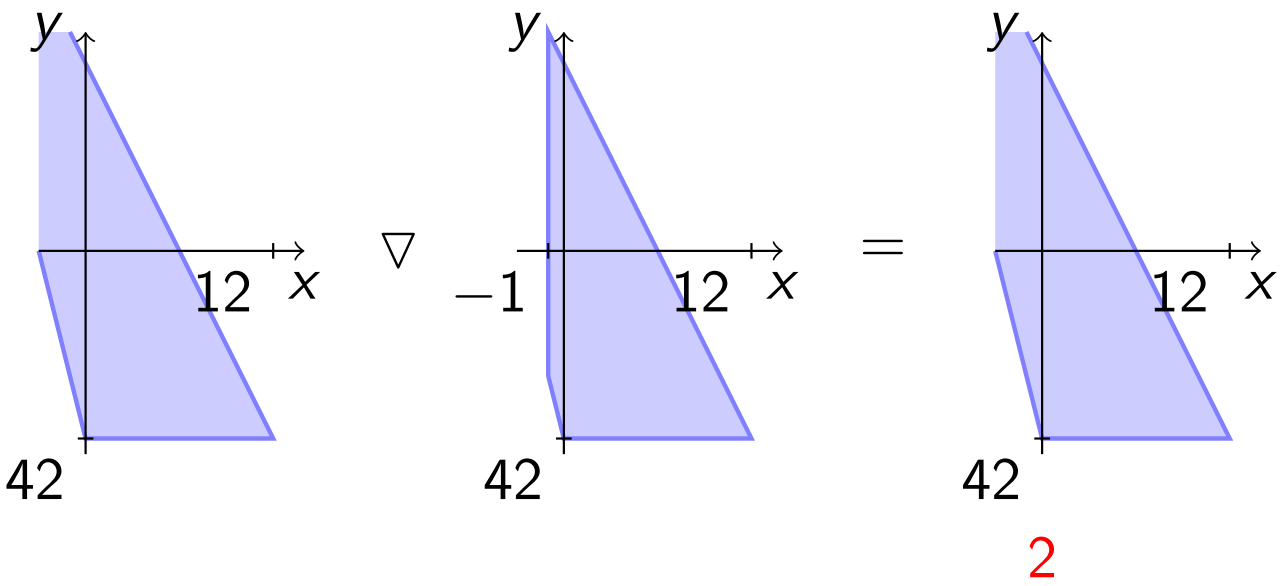
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

**}**$_5$



# Exemple (suite)

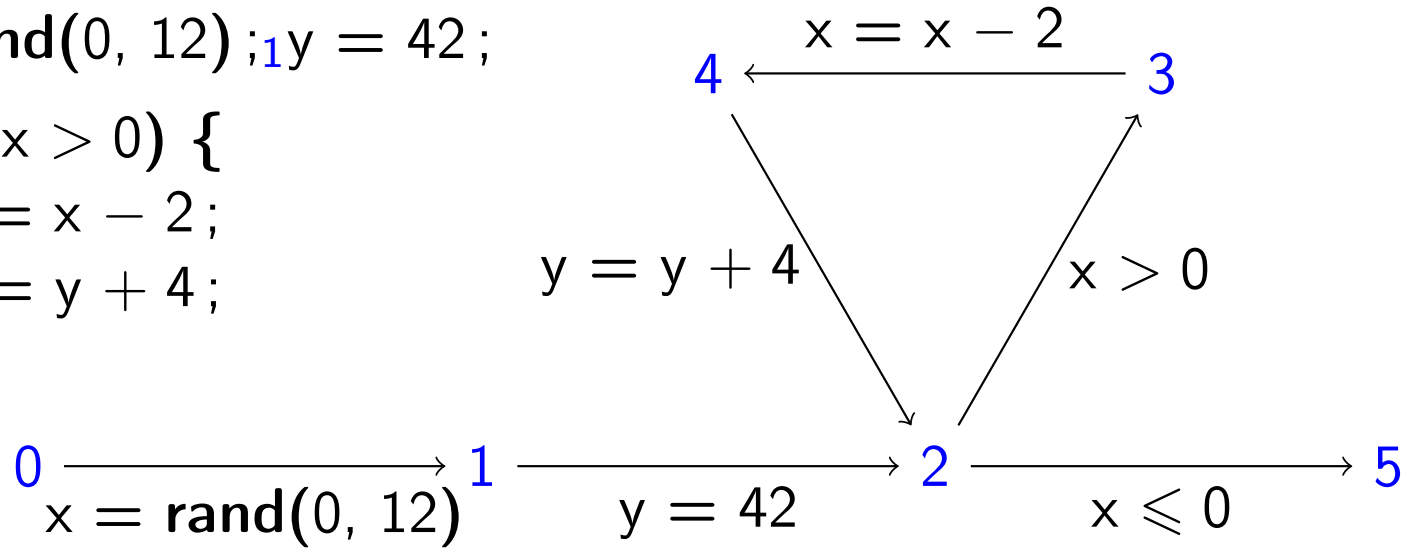
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

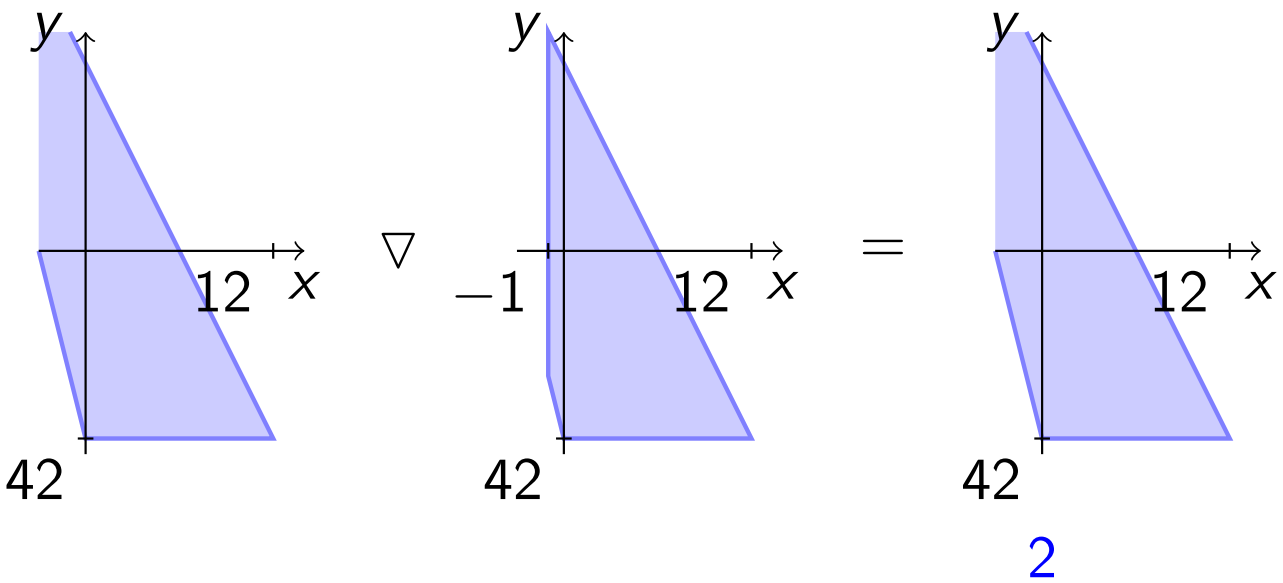
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

**}**$_5$




point fixe atteint

# Exemple (suite)

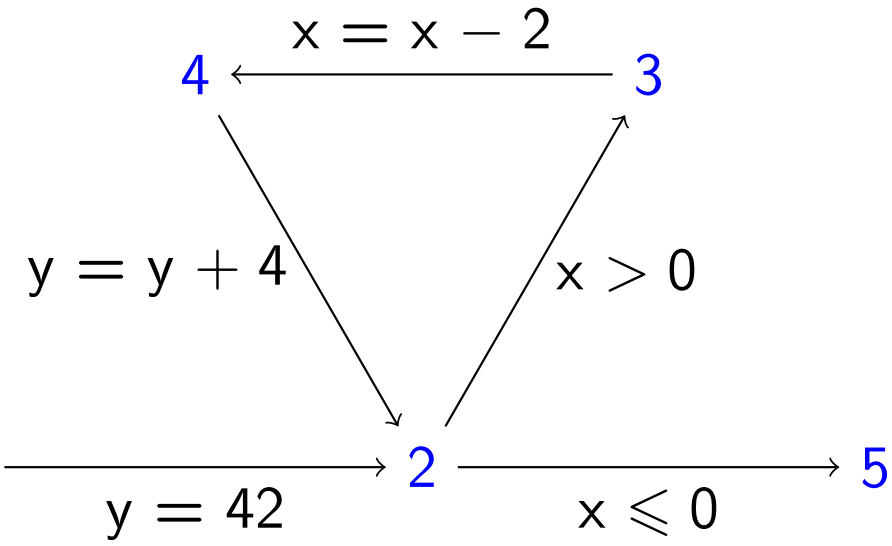
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

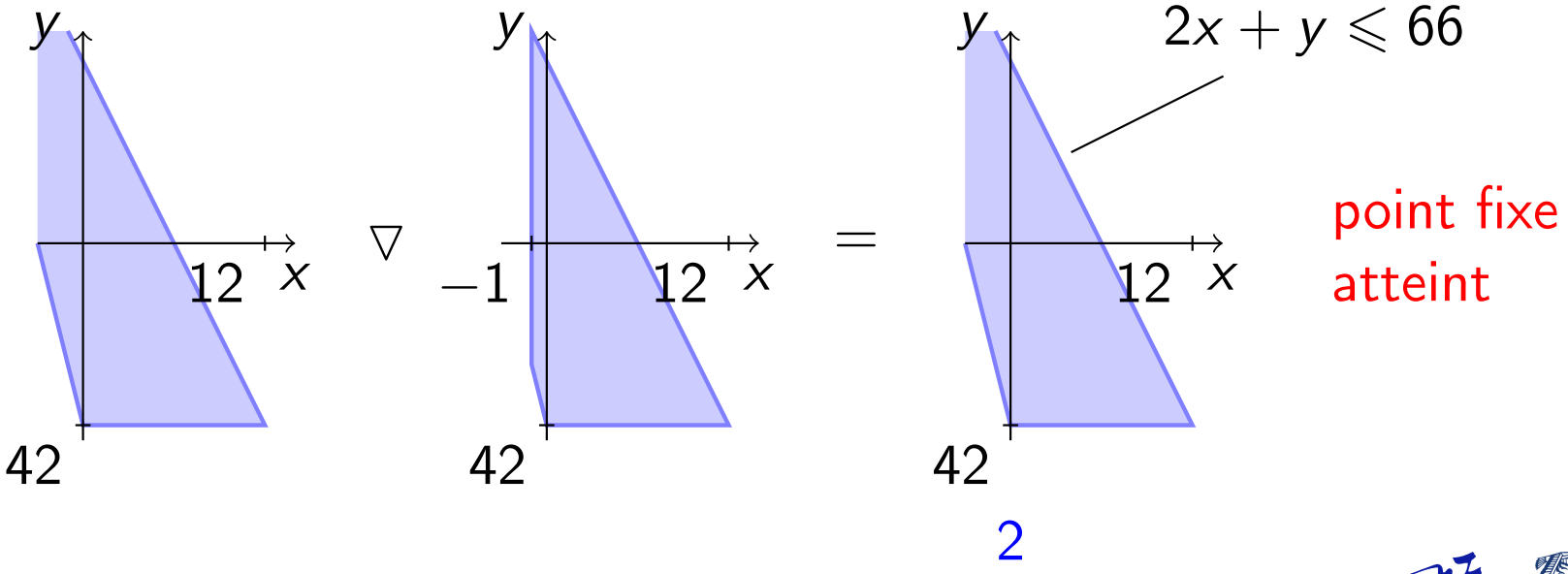
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

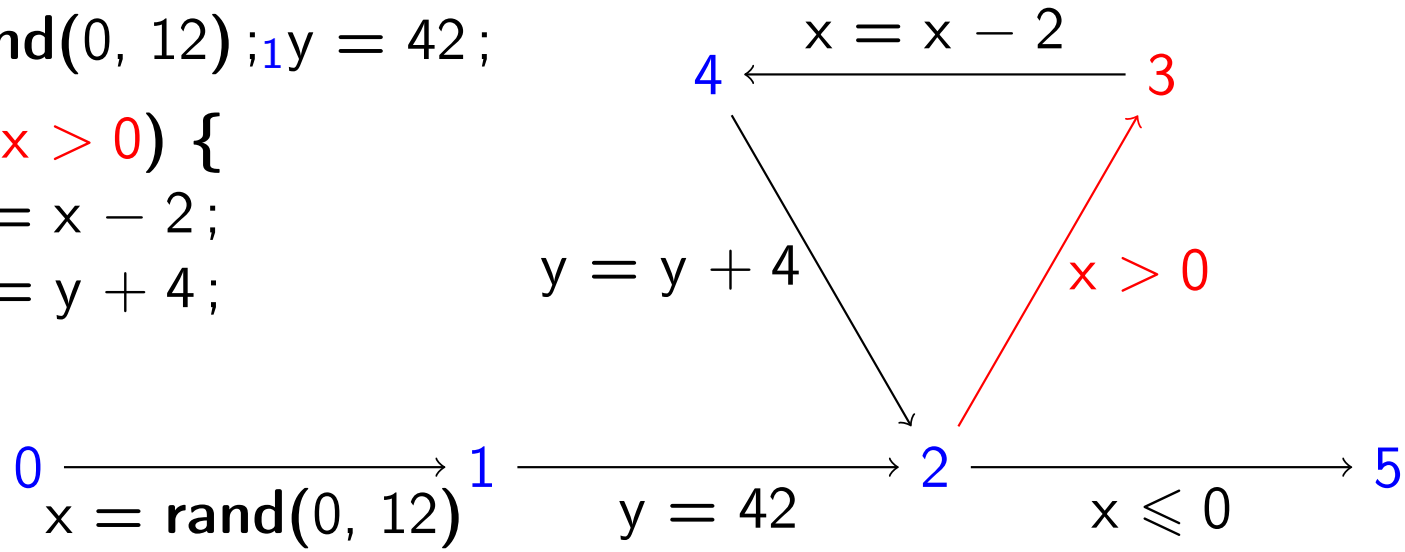
**}**$_5$

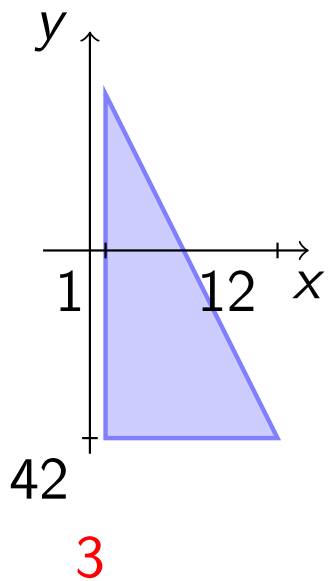


# Exemple (suite)

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) {
    $_3$x = x − 2 ;
    $_4$y = y + 4 ;
}$_5$



# Exemple (suite)

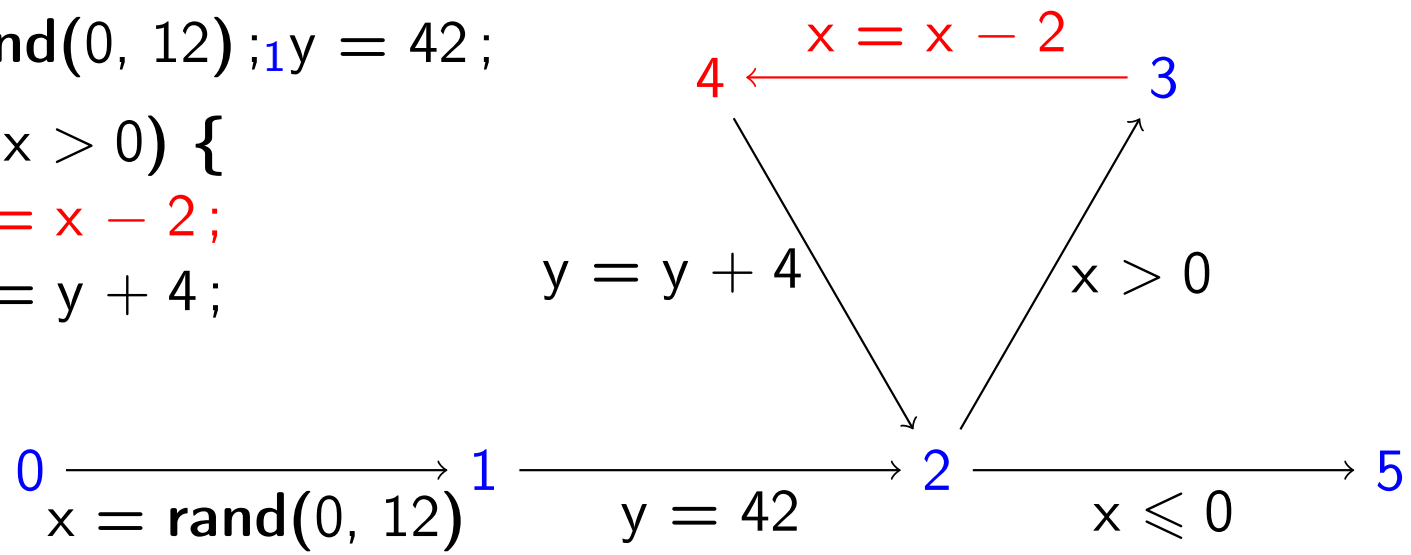
$_0x = \mathbf{rand}(0, 12)\,;{}_1y = 42\,;$

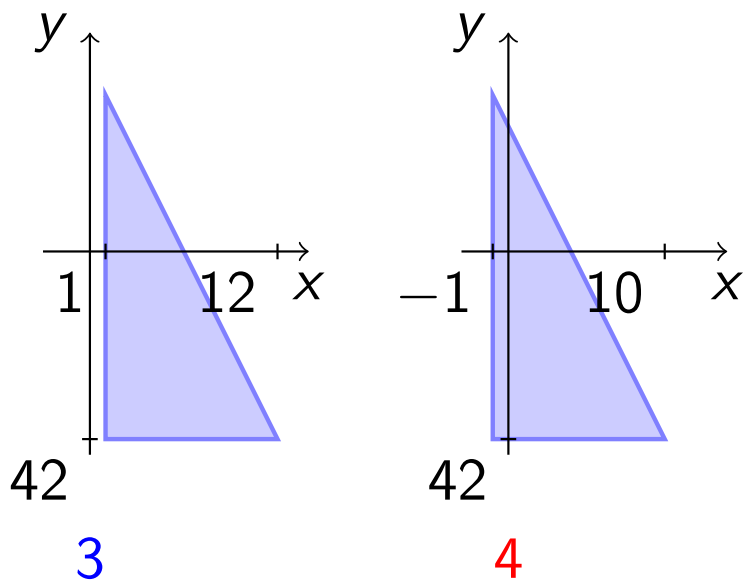
**while** $_2(x > 0)$ **{**

$\quad {}_3x = x - 2\,;$

$\quad {}_4y = y + 4\,;$

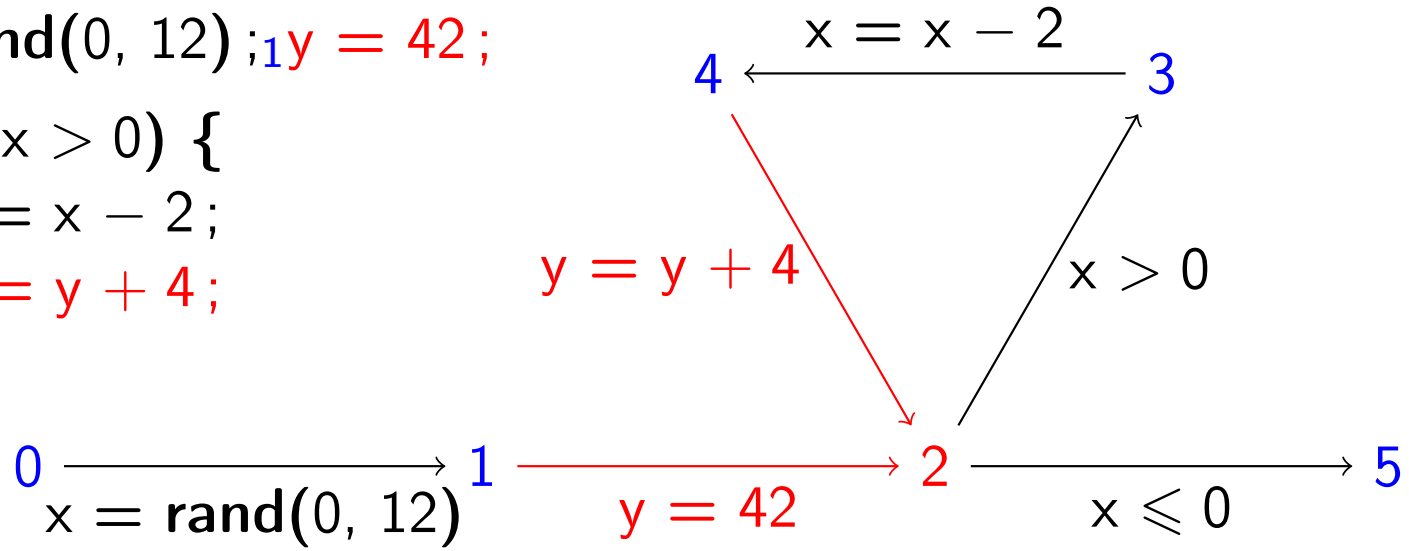
$\}_5$

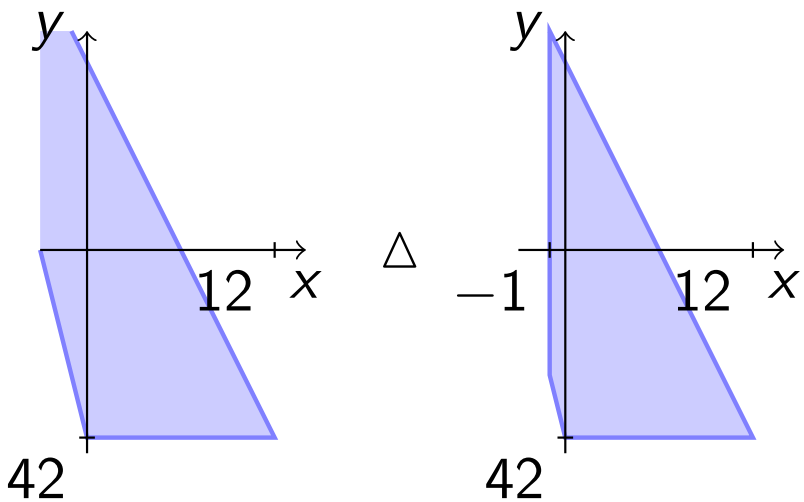


# Exemple (suite et fin)

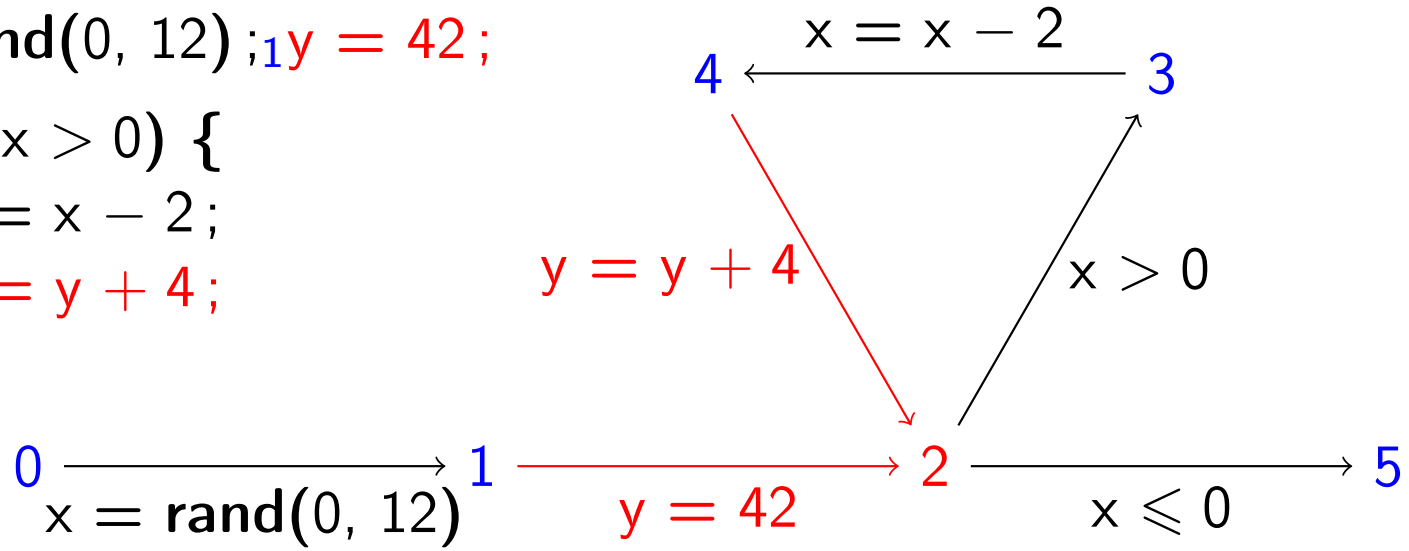
$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) {
    $_3$x = x − 2 ;
    $_4$y = y + 4 ;
}$_5$

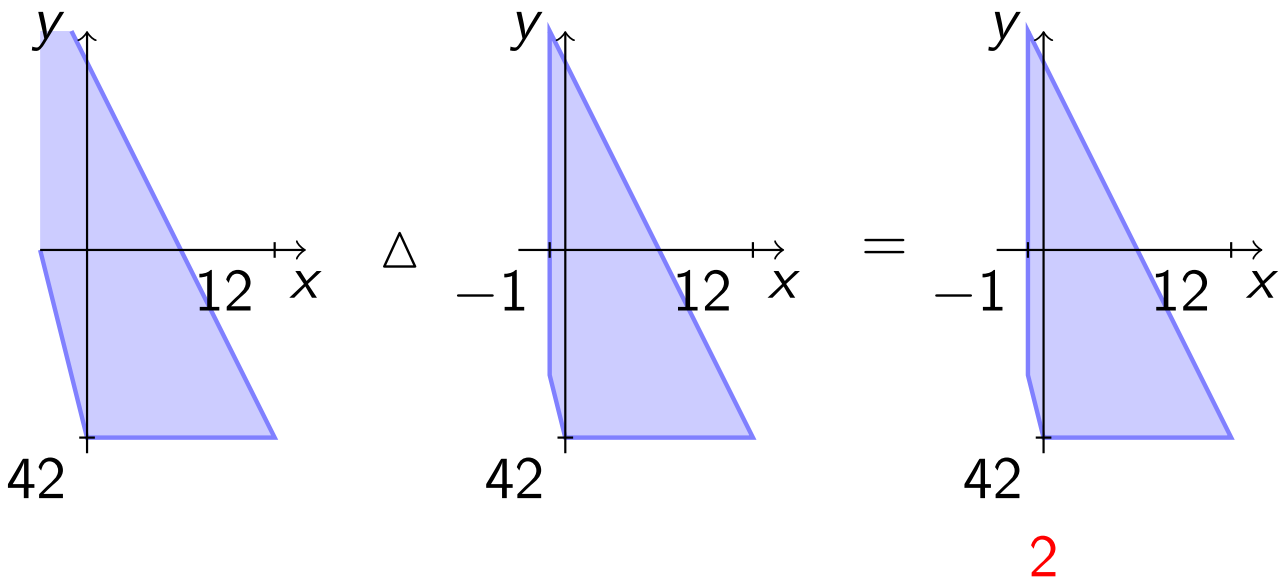


# Exemple (suite et fin)

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) {
    $_3$x = x − 2 ;
    $_4$y = y + 4 ;
}$_5$



# Exemple (suite et fin)

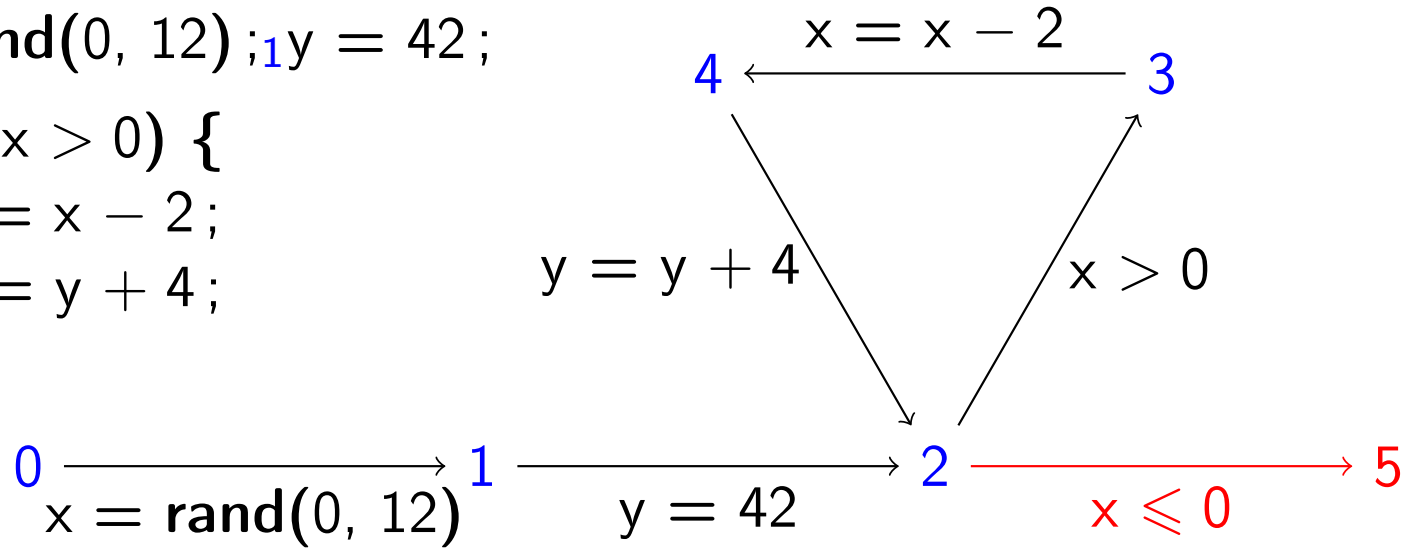
$_0$x = **rand**(0, 12) ;$_1$y = 42 ;

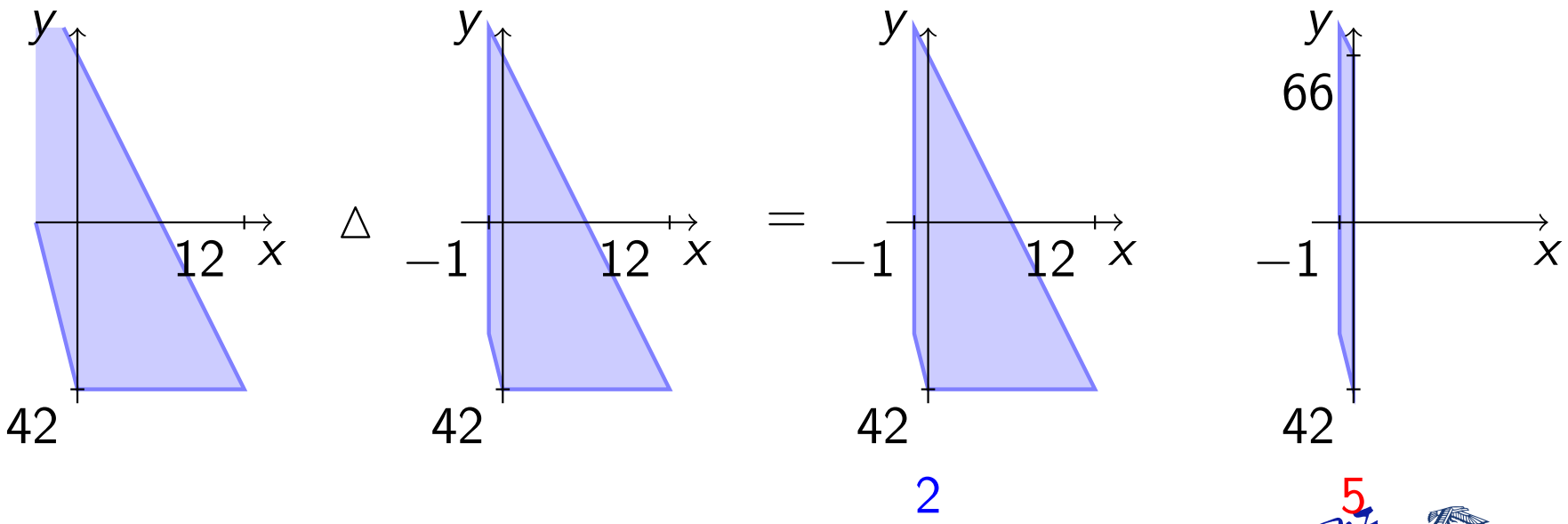
**while** $_2$(x > 0) {

  $_3$x = x − 2 ;

  $_4$y = y + 4 ;

}$_5$

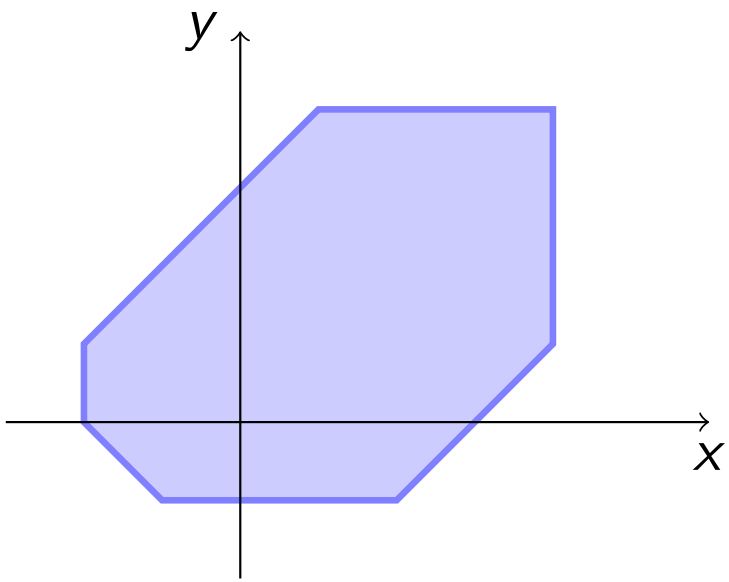


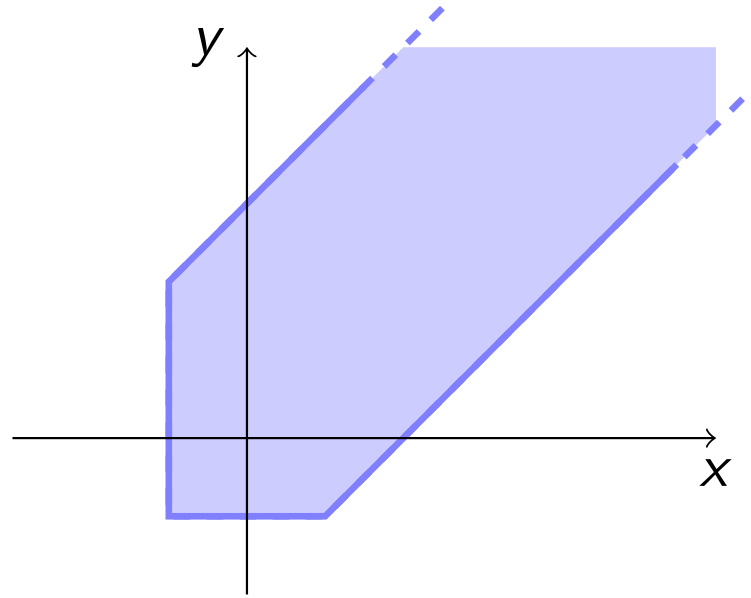




2 5

# Octogones

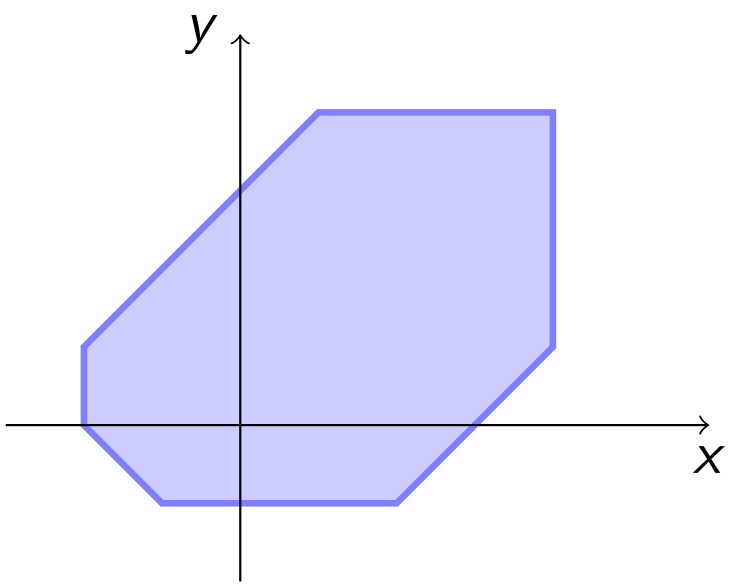
Similaire aux polyèdres mais en autorisant seulement les pentes multiples de 45°.

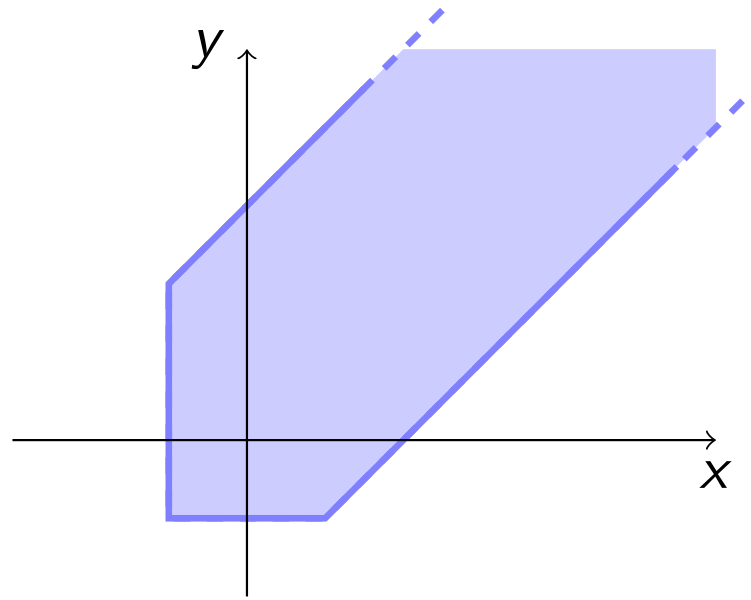


# Octogones

Similaire aux polyèdres mais en autorisant seulement les pentes multiples de 45°.




- − moins précis
- \+ meilleure complexité : chaque opération est en $\mathrm{O}\left(n^3\right)$ (complexité au pire cas exponentielle pour les polyèdres)

# Octogones, exercice

## Exercice 2

On considère le programme suivant :

```
0x = rand(0, 12);1y = 0;
while 2(x > 0) {
    3if (rand(0, 1) > 0) {
        4x = x − 1;
    } else {
        5x = x − 2;
    }
    6y = y + 1;
}7
```

1. Dessiner le graphe de flot de contrôle.
2. Calculer le point fixe.
3. Le raffiner par une itération descendante (avec $\triangle$).

# Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

# Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

- égalités affines ($2x + 3y = 5$)

# Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

- égalités affines ($2x + 3y = 5$)
- congruences ($x + 2y$ congru à 3 modulo 5)

# Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

- égalités affines ($2x + 3y = 5$)
- congruences ($x + 2y$ congru à 3 modulo 5)
- polyèdres tropicaux (polyèdres sur une algèbre $(\max, +)$)
- ...

# Domaines non numériques

Tous les domaines abstraits ne sont pas numériques.

## Exemple (listes)

On peut abstraire une liste en retenant
si elle est vide (nil) ou non (non_nil).

# Domaines non numériques

Tous les domaines abstraits ne sont pas numériques.

## Exemple (listes)

On peut abstraire une liste en retenant
si elle est vide (nil) ou non (non_nil).

Exemple : concaténation de deux listes

| @ | nil | non_nil |
|---|---|---|
| nil | nil | non_nil |
| non_nil | non_nil | non_nil |

# Domaines non numériques

Tous les domaines abstraits ne sont pas numériques.

## Exemple (listes)

On peut abstraire une liste en retenant
si elle est vide (nil) ou non (non_nil).

Exemple : concaténation de deux listes

| @ | nil | non_nil |
|---|---|---|
| nil | nil | non_nil |
| non_nil | non_nil | non_nil |

Exemple d'utilisation : prouver qu'on n'essaye jamais d'acceder
à la tête d'une liste vide (`List.hd []` en Caml).

# Virgule flottante

- Les nombres réels ne sont pas représentable en machine.

# Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.

# Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).

# Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).
- Problème : comment abstraire correctement ces arrondis.

# Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).
- Problème : comment abstraire correctement ces arrondis.

Solutions :

- pour les intervalles : arrondir les bornes vers l'extérieur ;

# Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).
- Problème : comment abstraire correctement ces arrondis.

Solutions :

- pour les intervalles : arrondir les bornes vers l'extérieur ;
- plus généralement : on peut abstraire une opération flottante $\mathrm{round}(a+b)$ par une opération réelle $(1+\epsilon)(a+b)$ puis utiliser des domaines sur les réels ;
- reste alors à implémenter correctement des domaines sur les réels, c'est un autre problème (on peut utiliser des rationnels par exemple).

# Partitionnement, exemple

```
0x = rand(-12, 12);
1if (x > 0) {
      2x = x + 1;
} else {
      3x = x - 1;
}
4y = 1 / x;5
```

# Partitionnement, exemple

```
0x = rand(-12, 12);
1if (x > 0) {
      2x = x + 1;
} else {
      3x = x - 1;
}
4y = 1 / x;5
```

- Après 2, on a a $x \in [\![2, 13]\!]$
- Après 3, on a a $x \in [\![-13, -1]\!]$

# Partitionnement, exemple

```
0 x = rand(-12, 12) ;
1 if (x > 0) {
      2 x = x + 1 ;
} else {
      3 x = x - 1 ;
}
4 y = 1 / x ;5
```

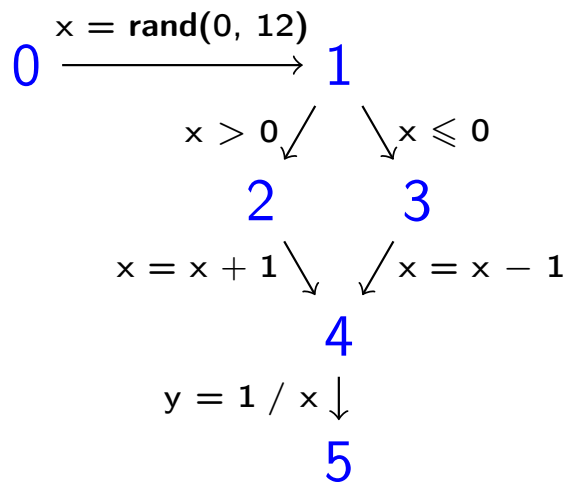
- Après 2, on a a $x \in [\![2, 13]\!]$
- Après 3, on a a $x \in [\![-13, -1]\!]$
- D'où en 4, $x \in [\![-13, 13]\!]$
  et une fausse alarme division par 0

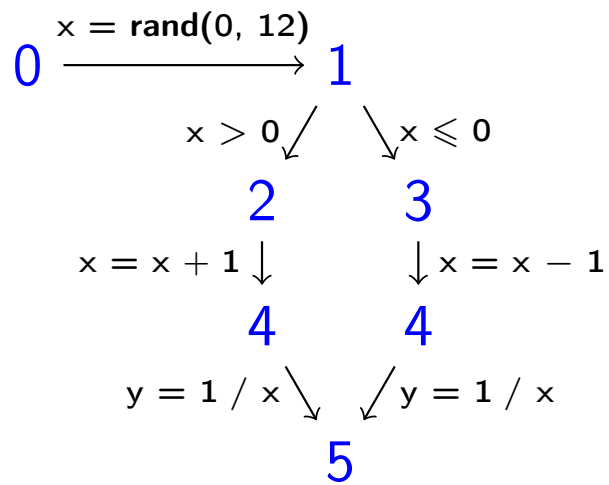
# Partitionnement, exemple

```
0x = rand(-12, 12);
1if (x > 0) {
      2x = x + 1;
} else {
      3x = x - 1;
}
4y = 1 / x;5
```

- Après 2, on a a $x \in [\![2, 13]\!]$
- Après 3, on a a $x \in [\![-13, -1]\!]$
- D'où en 4, $x \in [\![-13, 13]\!]$
  et une fausse alarme division par 0

Solution : déplacer le calcul de la borne supérieure des intervalles après l'affectation y := 1 / x.



# Stratégies d'itération

- Le widening/narrowing marche plutôt bien.

# Stratégies d'itération

- Le widening/narrowing marche plutôt bien.
- Mais il est difficile de concevoir un bon widening.

# Stratégies d'itération

- Le widening/narrowing marche plutôt bien.
- Mais il est difficile de concevoir un bon widening.
- D'où l'intérêt pour d'autres méthodes d'itération :
  - accélération ;
  - itération sur les stratégies (policy iteration).

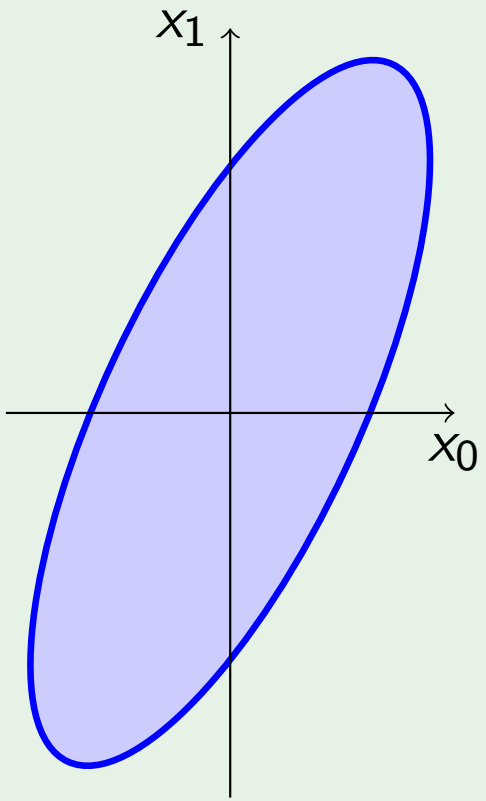
# Invariants quadratiques

- Certains systèmes n'ont pas de bon invariant linéaire mais ont de bons invariants quadratiques (ellipsoïdes).

# Invariants quadratiques

- Certains systèmes n'ont pas de bon invariant linéaire mais ont de bons invariants quadratiques (ellipsoïdes).
- On peut calculer ce genre d'invariants avec des outils d'optimisation (programmation semi définie).

## Exemple




```
node coupled_mass(u0, u1 : real)
returns (x0, x1, x2, x3 : real)
let
  assert(u0 >= -1.0 and u0 <= 1.0);
  assert(u1 >= -1.0 and u1 <= 1.0);
  x0 = 0.0->0.6227*pre(x0)+0.3871*pre(x1)
    -0.1130*pre(x2)+0.0102*pre(x3)
    +0.3064*pre(u0)+0.1826*pre(u1);
  x1 = 0.0->-0.3407*pre(x0)+0.9103*pre(x1)
    -0.3388*pre(x2)+0.0649*pre(x3)
    -0.0054*pre(u0)+0.6731*pre(u1);
  x2 = 0.0->0.0918*pre(x0)-0.0265*pre(x1)
    -0.7319*pre(x2)+0.2669*pre(x3)
    -0.0494*pre(u0)+1.6138*pre(u1);
  x3 = 0.0->0.2643*pre(x0)-0.1298*pre(x1)
    -0.9903*pre(x2)+0.3331*pre(x3)
    -0.0531*pre(u0)+0.4012*pre(u1);
tel
```

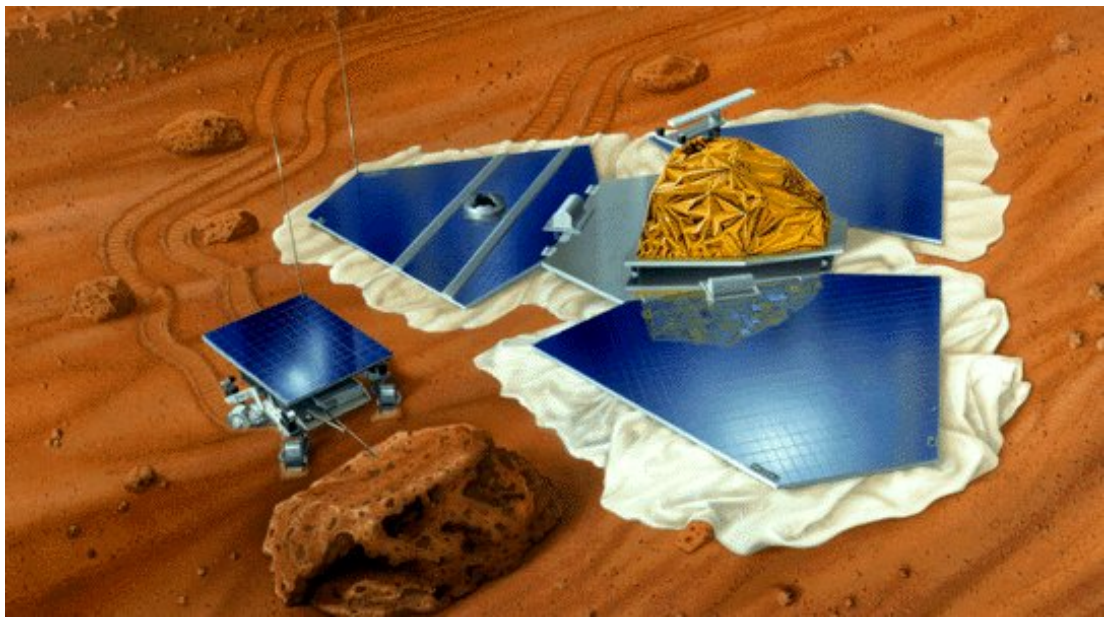
# Astrée

- Développé par l'équipe de Patrick Cousot à l'ÉNS Ulm.
- Preuve d'absence d'erreur à l'exécution
  dans du code temps réel embarqué.
- Utilisé pour les commandes de vol des Airbus
  (plusieurs centaines de milliers de lignes de C).
- Commercialisé par AbsInt GmbH


`http://www.astree.ens.fr/`

# IKOS : Inference Kernel for Open Static Analyzers

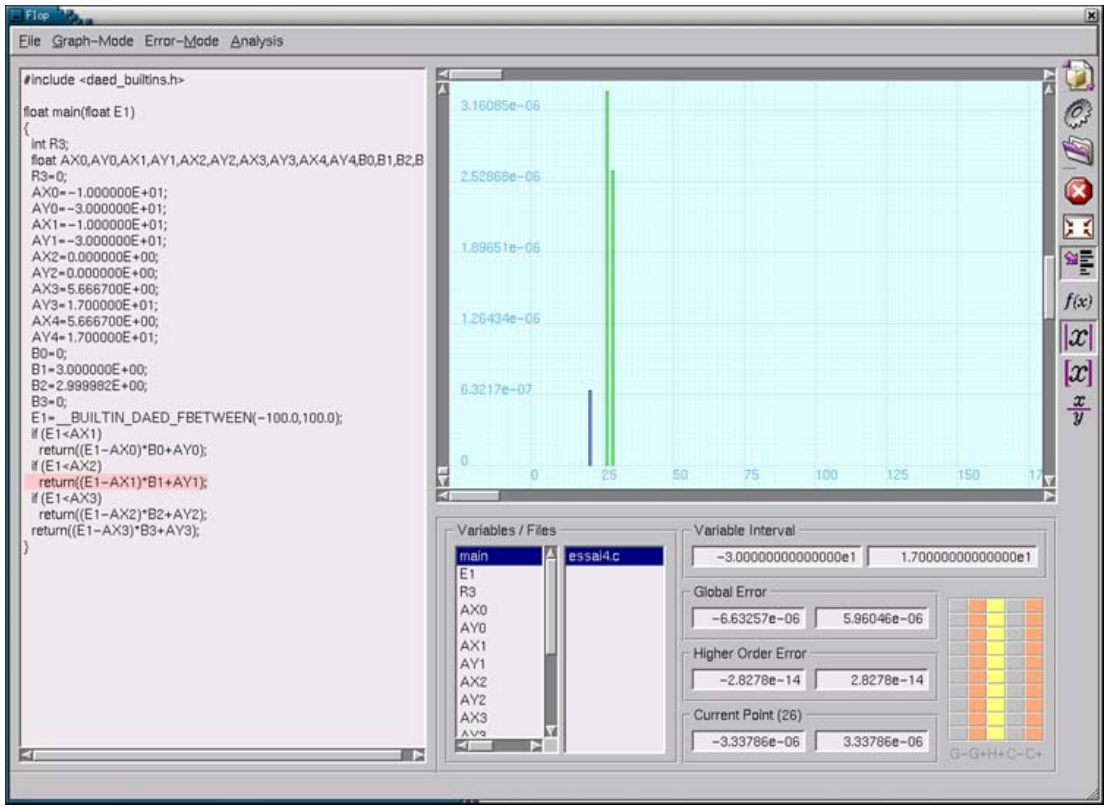
- Développé par la NASA. Open-Source
- Objectifs similaires à Astrée. Cible C++
- Prédécesseur (CGS) utilisé sur les contrôleurs de vols de : Mars Pathfinder, Deep Space One,...



`http://ti.arc.nasa.gov/opensource/ikos/`
`https://github.com/NASA-SW-VnV/ikos`
Développé par Arnaud Venet et Maxime Arthaud (N7)

# Fluctuat

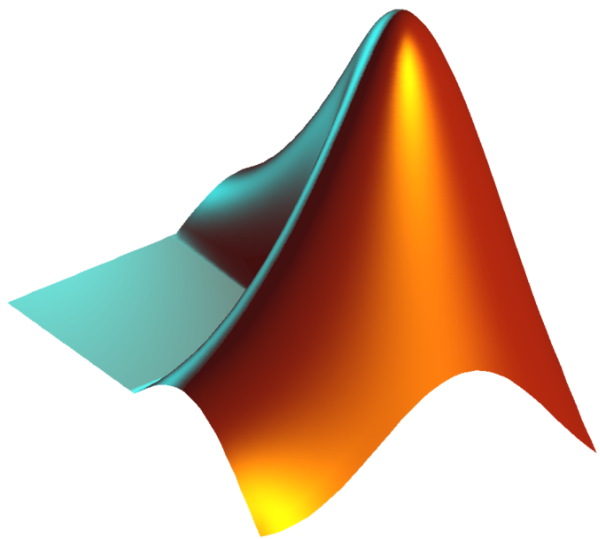
- Développé par l'équipe d'Éric Goubault au CEA.
- Analyse des erreurs d'arrondi en virgule flottante.
- Utilisé par divers industriels.




`http://www-list.cea.fr/labos/fr/LSL/fluctuat/index.html`

# Polyspace

- Vendu par MathWorks.
- Plus généraliste.
- Moins de possibilité de réglage des domaines abstraits
- Utilisé par divers industriels.



`http://www.polyspace.com/`

# EVA – Evolved Value Analysis de Frama-C

Plugin d'analyse statique de la plateforme Frama-C

- VA – Value Analysis
- EVA - Evolved Value Analysis
  - Différents choix de modèles mémoires
  - Nombreux domaines abstraits disponibles

Open-source

`https://frama-c.com/value.html`

# Apron

- Librairie de domaines relationnels
  développée par Bertrand Jeannet (INRIA Rhône-Alpes)
  et Antoine Miné (CNRS, ÉNS).
- Polyèdres.
- Octogones.
- Implémenté en C.
- Interface en OCaml.

`http://apron.cri.ensmp.fr/library/`

# Liste des exercices

1. Affectaction avec la représentation par contraintes dans le domaine abstrait des polyèdres convexes
2. Analyse de l'exemple (graphe de flot de contrôle, calcul du post-point fixe, rétrécissement par itérations décroissante)