# Machines à compteurs:

## de la calculabilité à la vérification de programmes

Michael Blondin

# Introduction

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```

**Vérification**

# Introduction

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```

↓

## Vérification

**Le logiciel fait-il ce à quoi on s'attend?**

# Introduction

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```

**Vérification**

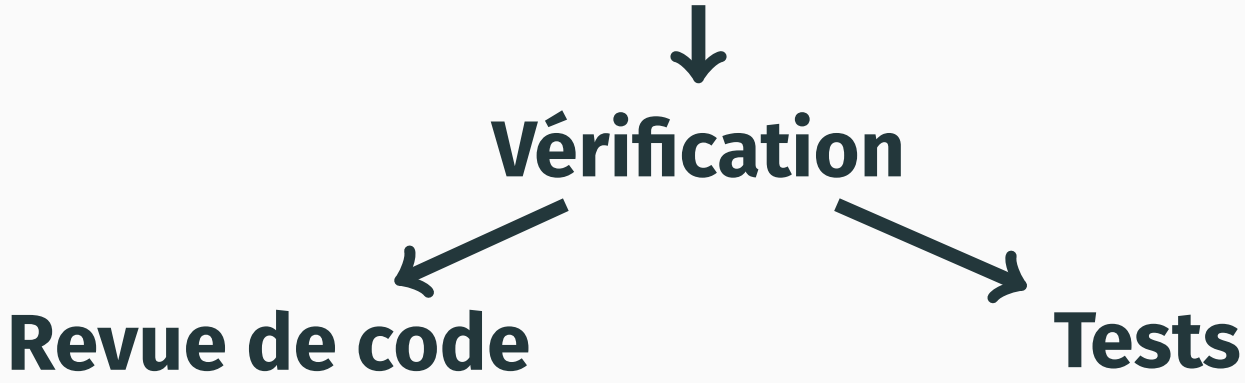
**30% à 50% du temps de développement**

# Introduction

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```

# Introduction

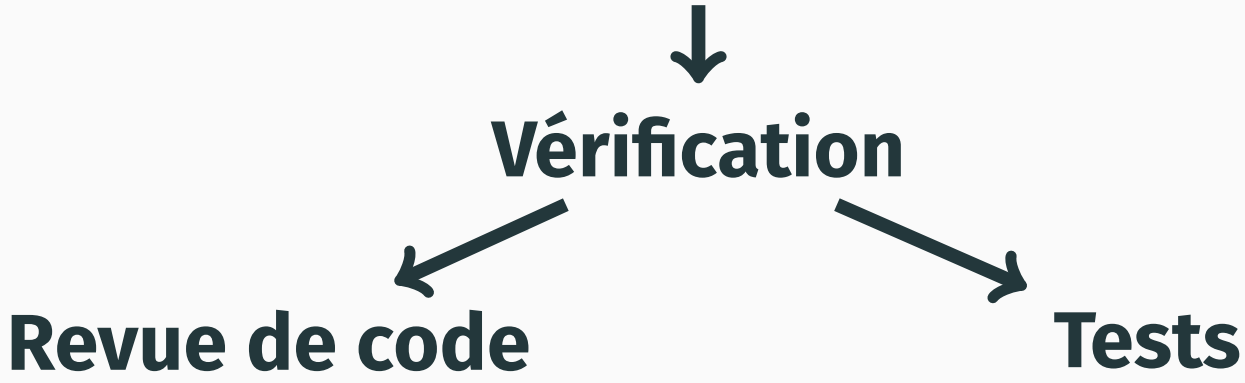
```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```




**Corrige la majorité des bogues...**

# Introduction

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```




**Corrige la majorité des bogues…**

**Mais ne peut pas montrer leur absence**

**et trouve les bogues « prévisibles par un humain »**

# Introduction

**Difficile d’anticiper les erreurs des systèmes concurrents**

# Introduction

## Difficile d'anticiper les erreurs des systèmes concurrents

**Mars Pathfinder**

**Mission compromise (1997)**

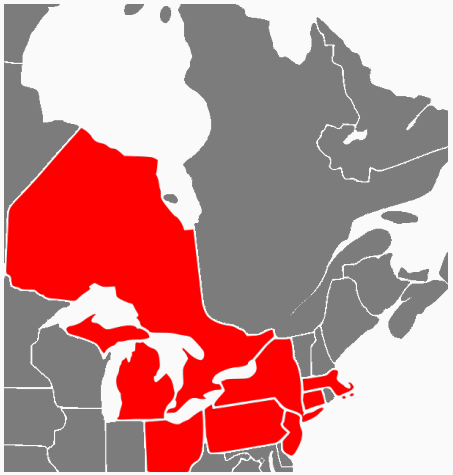
# Introduction

## Difficile d'anticiper les erreurs des systèmes concurrents

**Mars Pathfinder**

**Mission compromise (1997)**

**Réseau électrique**



**Panne généralisée (2003)**
**10 millions en Ontario**
**45 millions aux É.-U.**

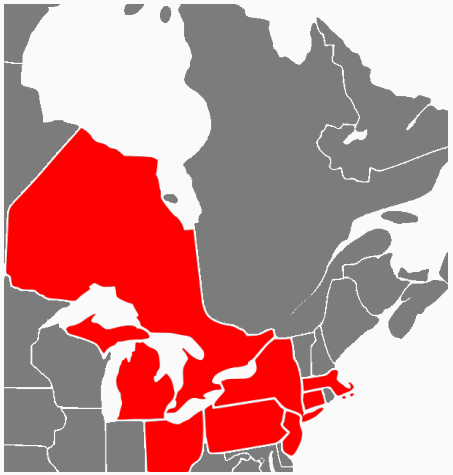
# Introduction

## Difficile d'anticiper les erreurs des systèmes concurrents

### Mars Pathfinder

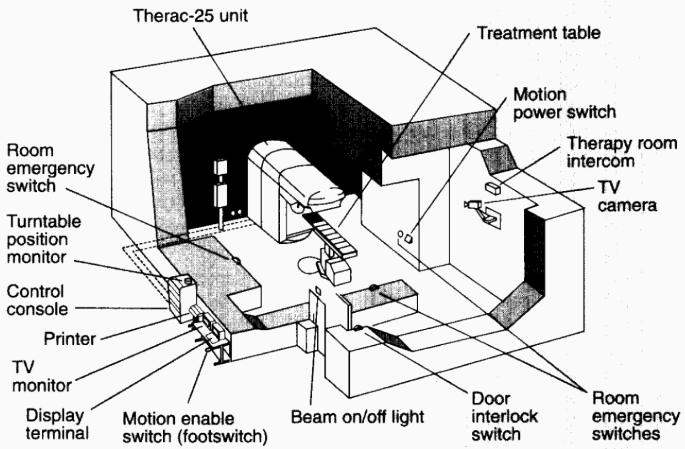
**Mission compromise (1997)**

### Réseau électrique



**Panne généralisée (2003)**
**10 millions en Ontario**
**45 millions aux É.-U.**

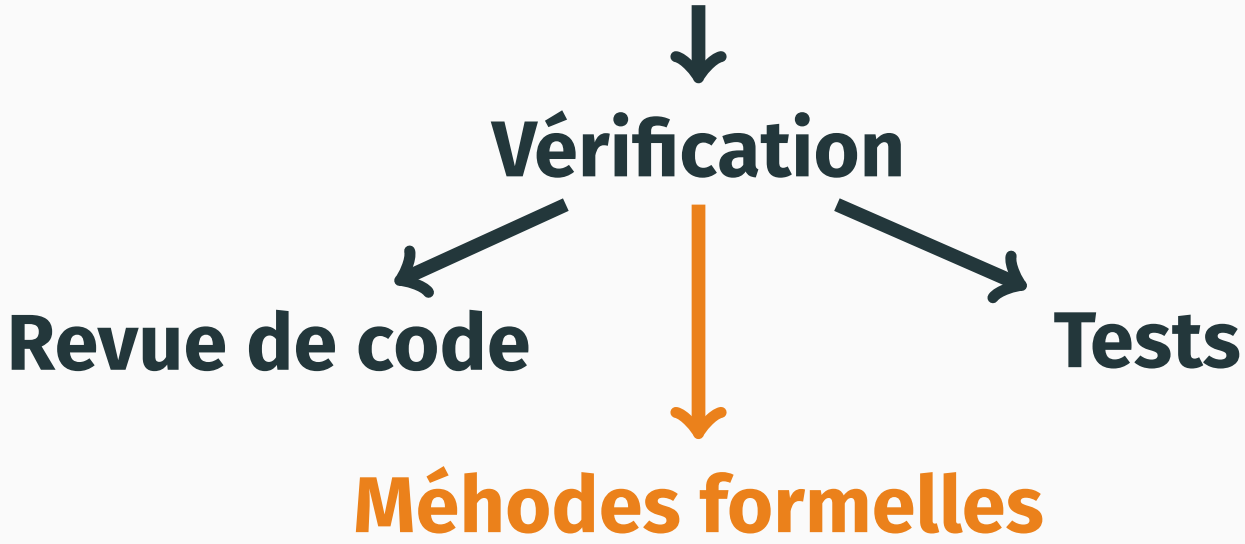
### Therac-25 (radiothérapie)




**Décès/blessures graves (1985-87)**
**6 personnes**

# Introduction

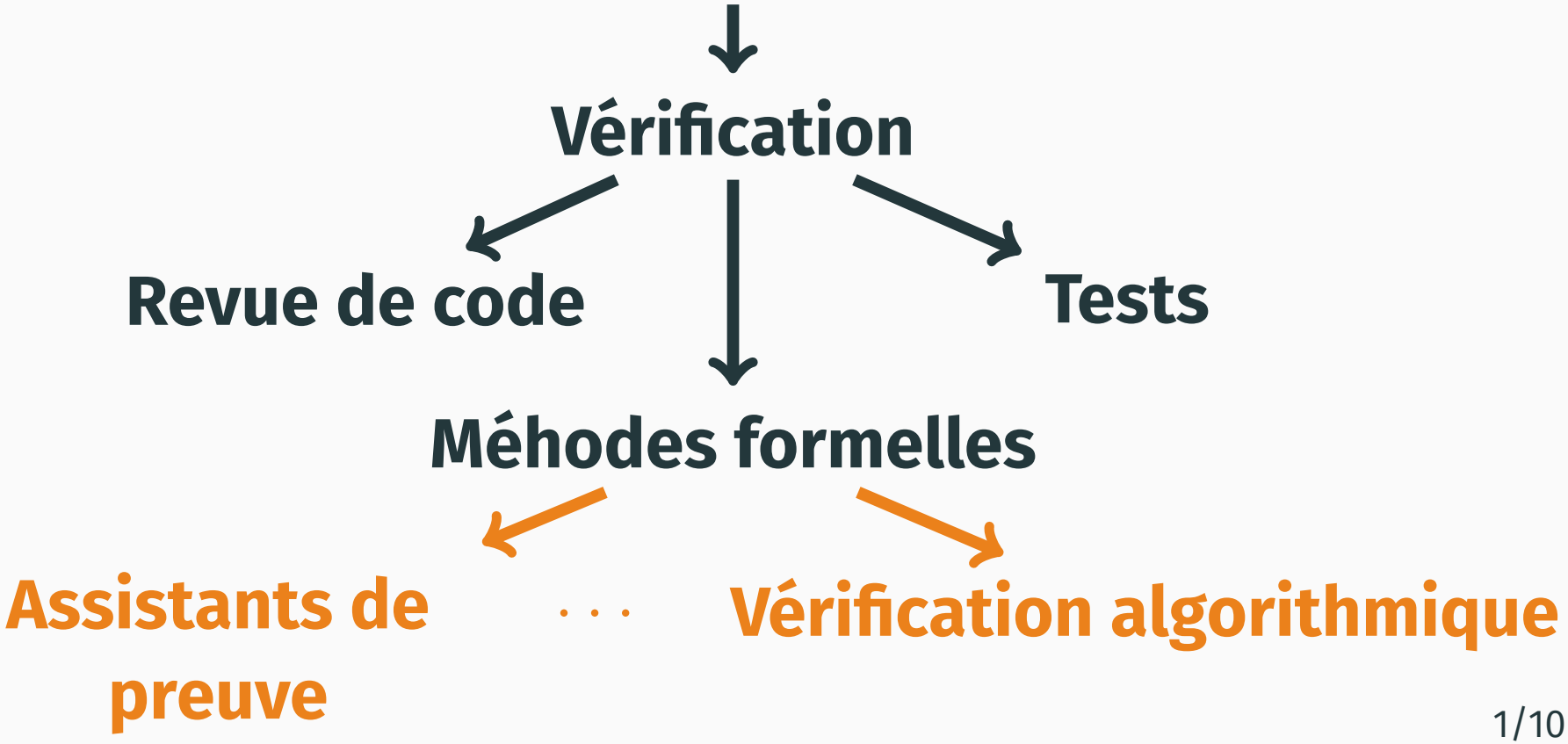
```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```




**Rigueur des mathématiques pour prouver correction de systèmes**

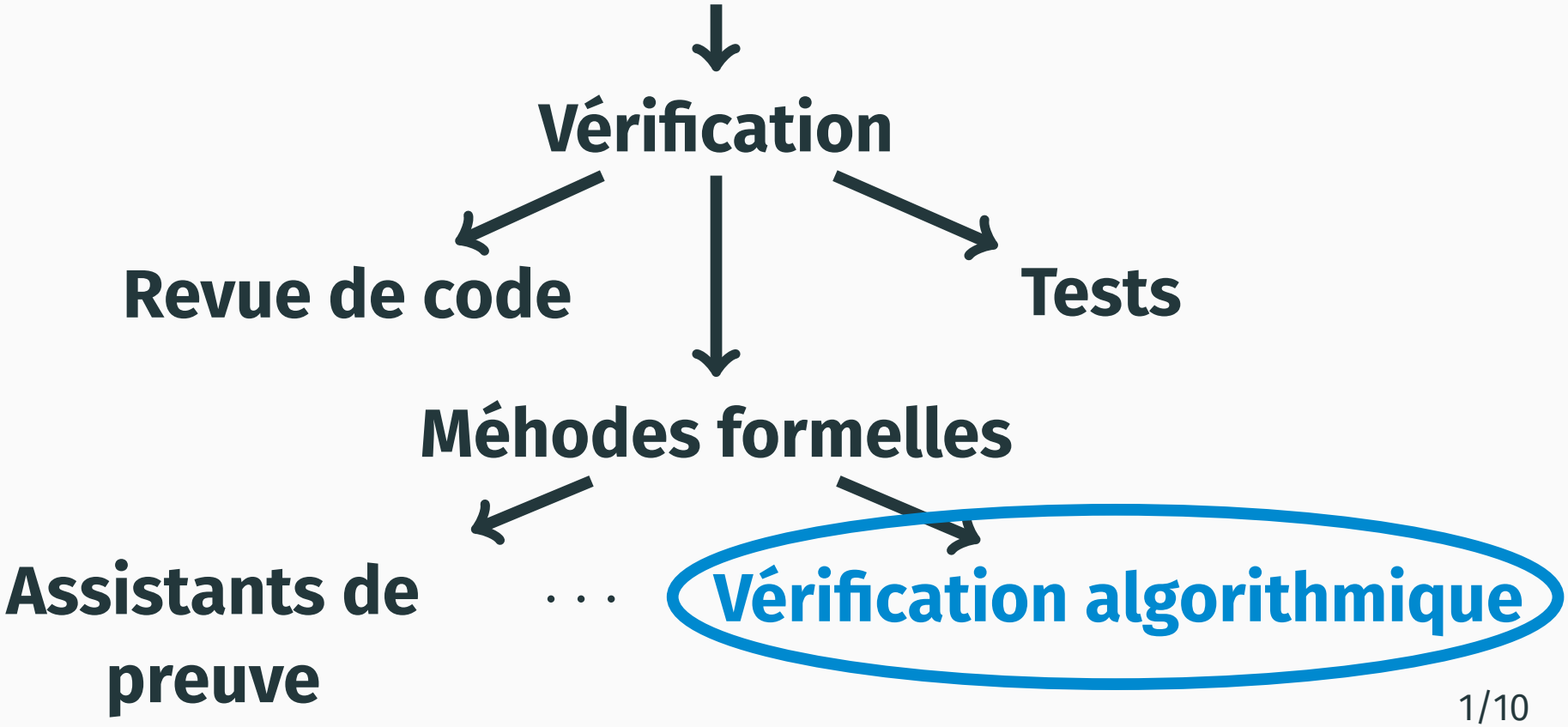
# Introduction

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```



# Introduction

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```

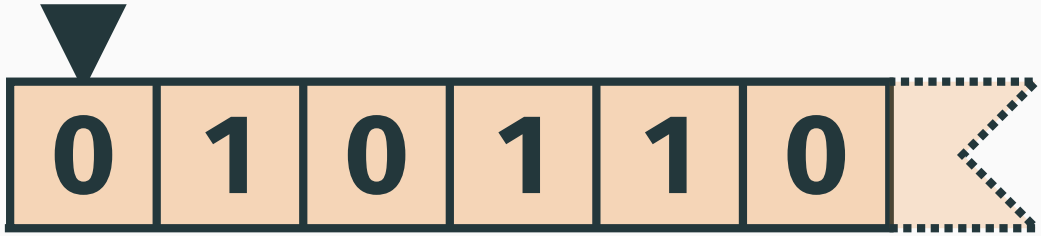


# Ordinateur et programmes

```
for (i = 0; i < 12; i++) {
  uint32_t keyval;
  char buf[3];
  if (OF_getprop(handle, key_names[i],
                 &keyval, sizeof(keyval)) < 0)
    continue;
  // ...
```

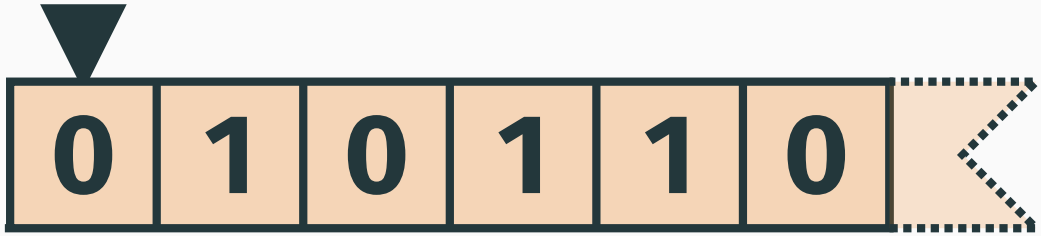
**Comment vérifier automatiquement qu'un programme est sans bogue?**

# Ordinateur et programmes



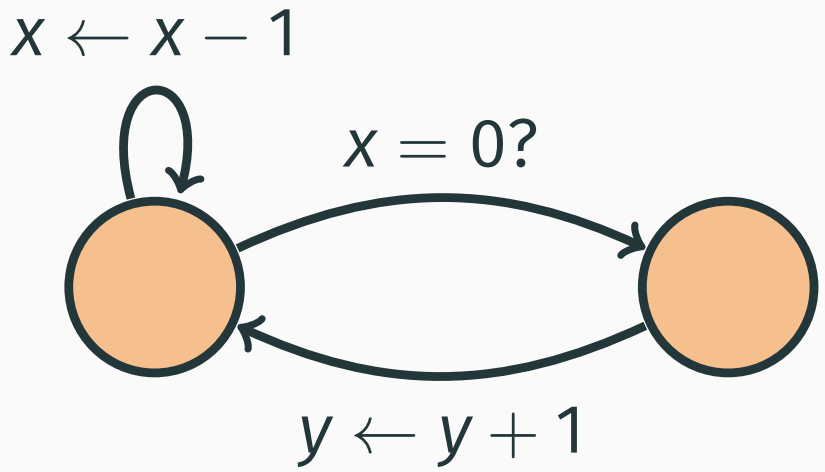

**On doit d'abord formaliser ordinateur et programmes**

# Ordinateur et programmes




**Machine de Turing, λ-calcul, automate à piles, machine à compteurs, etc.**

# Ordinateur et programmes




**Machine de Turing, λ-calcul, automate à piles, machine à compteurs, etc.**

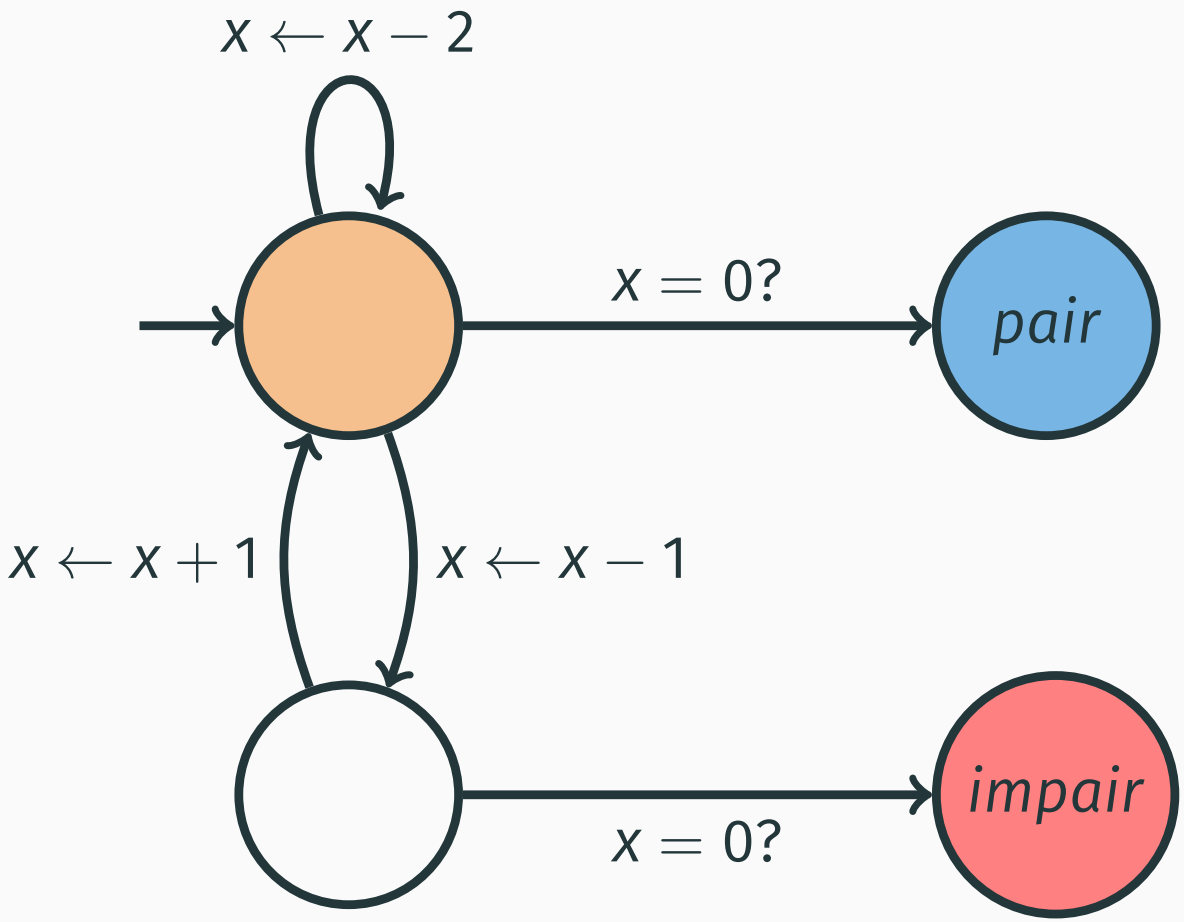
$\in \mathbb{N}$

# Plan de la présentation

1. **Machines à compteurs**
2. **Systèmes d'addition de vecteurs**
3. **Problème d'accessibilité**
4. **Extensions et problèmes ouverts**

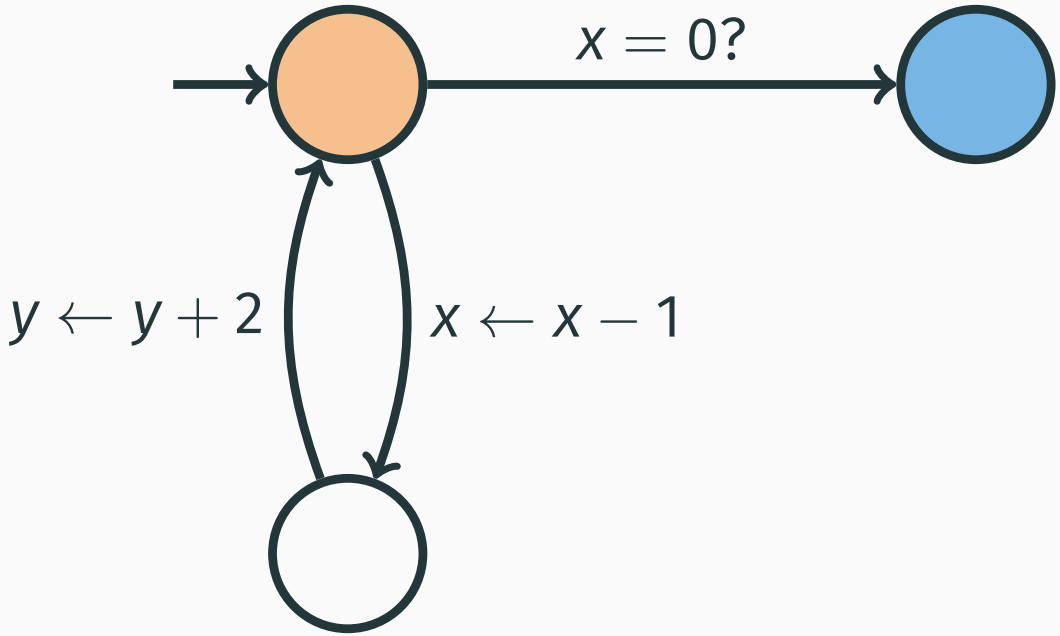
**Comment programmer une machine à compteurs?**

# Machines à compteurs



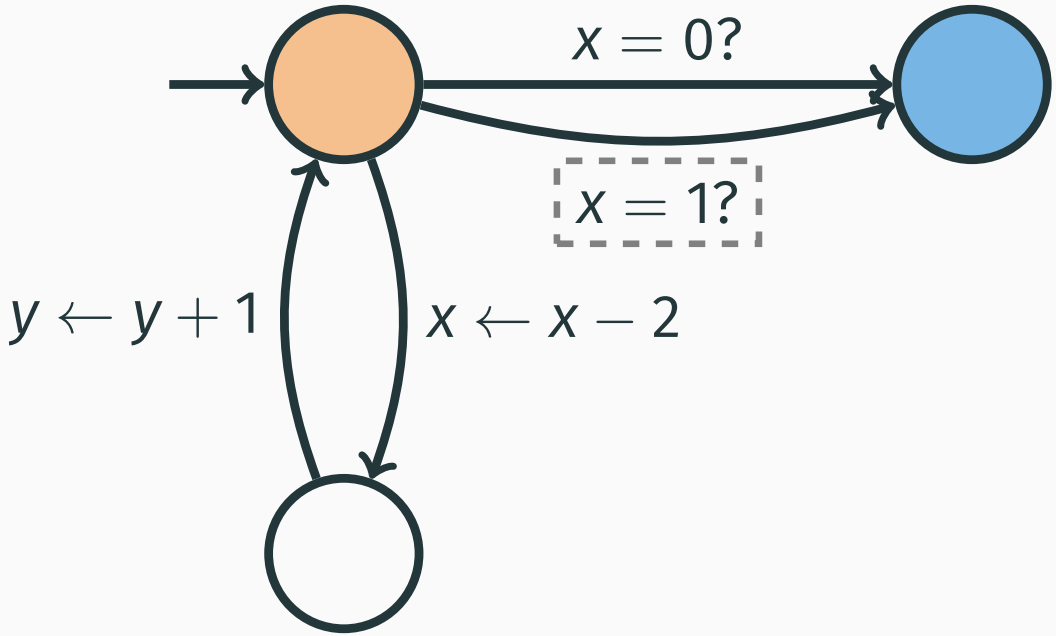

**Tester si $x$ est pair**

# Machines à compteurs



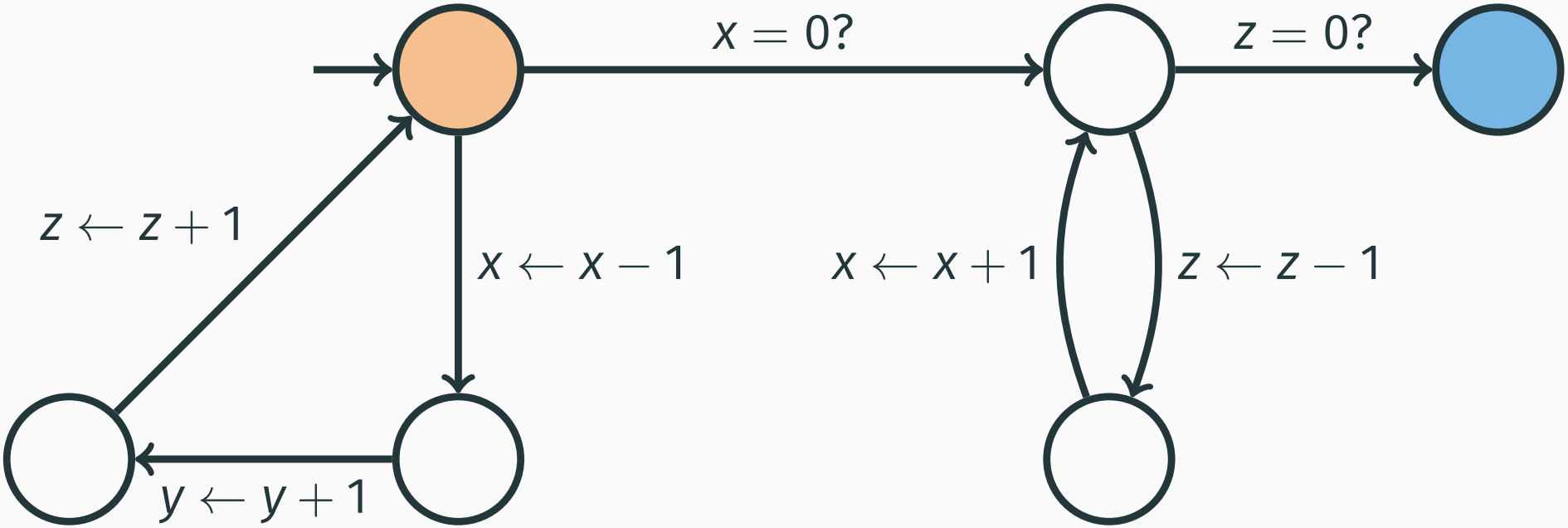

**Doubler $x$**

# Machines à compteurs




**Calculer** $x \div 2$

# Machines à compteurs




**Copier $x$ dans $y$**

On imagine la chaîne comme un nombre binaire

Par ex. $s =$ "1011" vaut $x = 2^3 + 2^1 + 2^0 = 13$

**Manipuler une chaîne de bits**

# Machines à compteurs

On imagine la chaîne comme un nombre binaire

Par ex. $s =$ "1011" vaut $x = 2^3 + 2^1 + 2^0 = 13$

- Examiner dernier bit: vérifier parité
- Ajouter bit $b$: doubler et additionner $b$
- Retirer dernier bit: diviser par 2
- Accès aux autres bits: copier puis restaurer

# Machines à compteurs

On représente: $a = 13, b = 100, c = 42$

par son codage de Gödel: $x = 2^{13} \cdot 3^{100} \cdot 5^{42}$

Un second compteur $y$ sert à extraire l'information

**Deux compteurs suffisent!**

# Vérification


compteurs = $\boldsymbol{u}$ compteurs = $\boldsymbol{v}$

**Comment détecter un bogue?**

# Vérification


compteurs = $\boldsymbol{u}$ compteurs = $\boldsymbol{v}$

**Indécidable: *aucun* algorithme ne résout ce problème ⚠**

# Vérification


compteurs = $\boldsymbol{u}$ compteurs = $\boldsymbol{v}$

**Théorème de Rice: *aucune* propriété (sémantique et non triviale) n'est décidable ⚠**

# Vérification

| | |
|---|---|
| Compteurs: | $x, y, \ldots \in \mathbb{N}$ |
| Incrémentation: | $x \leftarrow x + a$ |
| Décrémentation: | `si` $x \geq b$ `:` $x \leftarrow x - b$ |
| Non déterminisme: | `aller à` p `ou` q |
| Test à zéro: | `si` $x = 0$ `:` `aller à` p |

# Vérification

| | |
|---|---|
| Compteurs: | $x, y, \ldots \in \mathbb{N}$ |
| Incrémentation: | $x \leftarrow x + a$ |
| Décrémentation: | `si` $x \geq b$ `:` $x \leftarrow x - b$ |
| Non déterminisme: | `aller à` p `ou` q |
| ~~Test à zéro:~~ | ~~`si` $x = 0$ `: aller à` p~~ |

**Vérification possible!**

# Vérification

| | |
|---|---|
| Compteurs: | $x, y, \ldots \in \mathbb{N}$ |
| Incrémentation: | $x \leftarrow x + a$ |
| Décrémentation: | `si` $x \geq b$ `:` $x \leftarrow x - b$ |
| Non déterminisme: | `aller à` p `ou` q |

**Réseaux de Petri**

Petri

**Systèmes d'addition de vecteurs**

Karp et Miller

# Vérification

| | |
|---|---|
| Compteurs: | $x, y, \ldots \in \mathbb{N}$ |
| Incrémentation: | $x \leftarrow x + a$ |
| Décrémentation: | si $x \geq b$: $x \leftarrow x - b$ |
| Non déterminisme: | aller à p ou q |

**Réseaux de Petri**

Petri

**Systèmes d'addition de vecteurs**

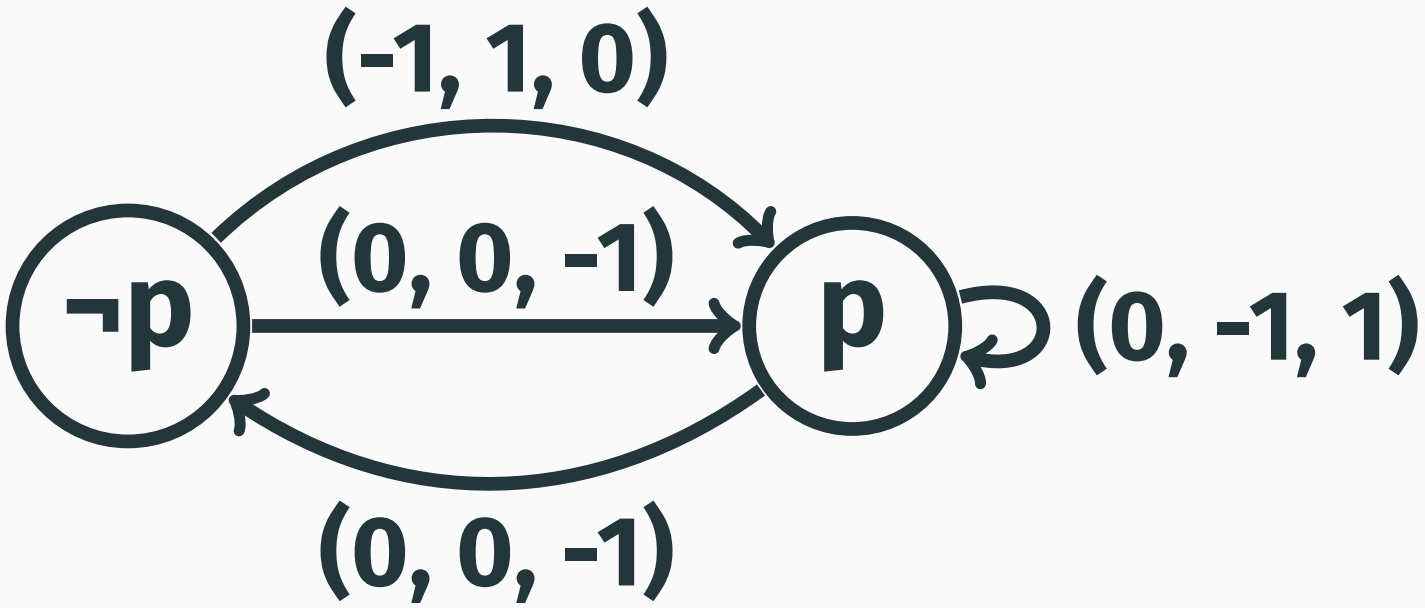
Karp et Miller

# Systèmes d'addition de vecteurs

**Cherchons à déterminer s'il est possible que tous les fils d'exécution terminent et que p = `vrai`**

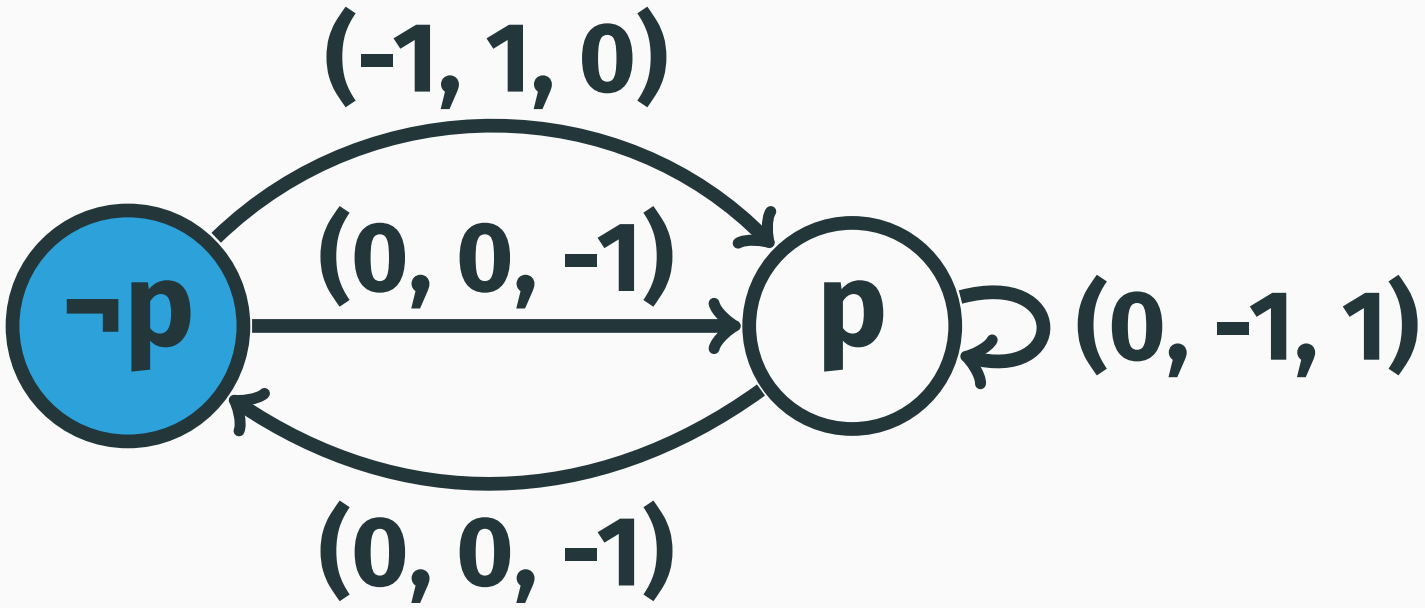
0. `si ¬p : p ← vrai, sinon: aller à 0`
1. `tant que ¬p: rien faire`
2. `p ← ¬p`

# Systèmes d'addition de vecteurs



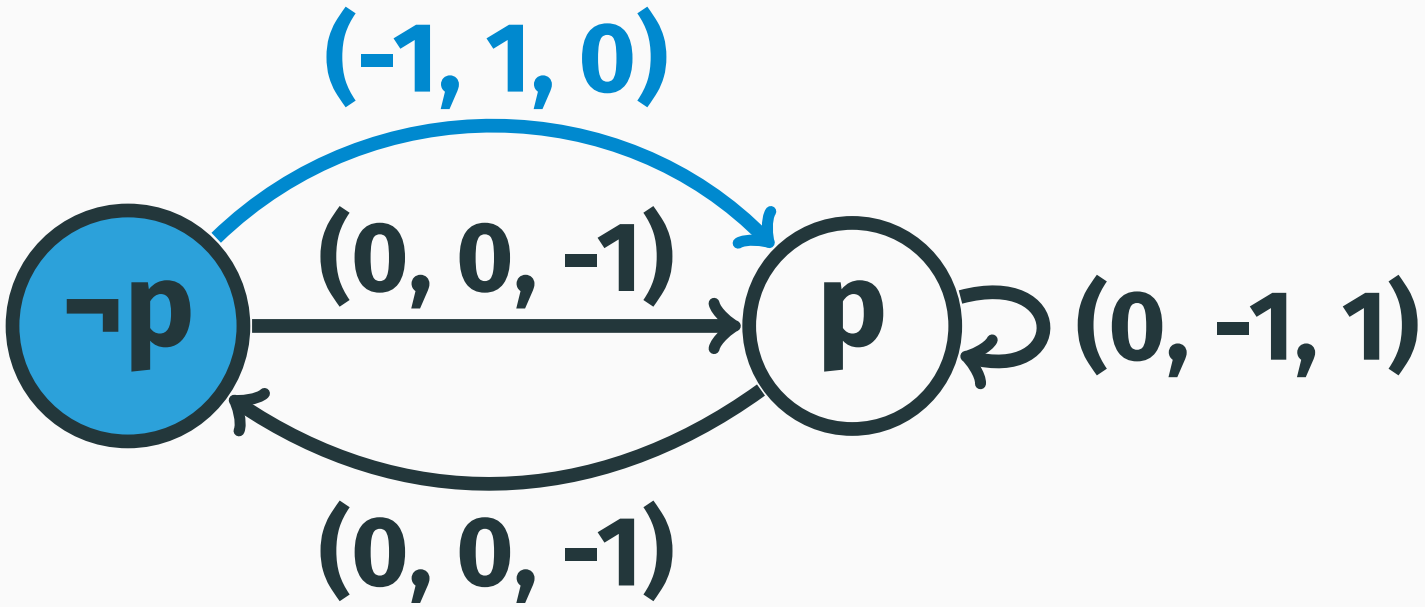

0. `si ¬p : p ← vrai, sinon: aller à 0`
1. `tant que ¬p: rien faire`
2. `p ← ¬p`

# Systèmes d'addition de vecteurs



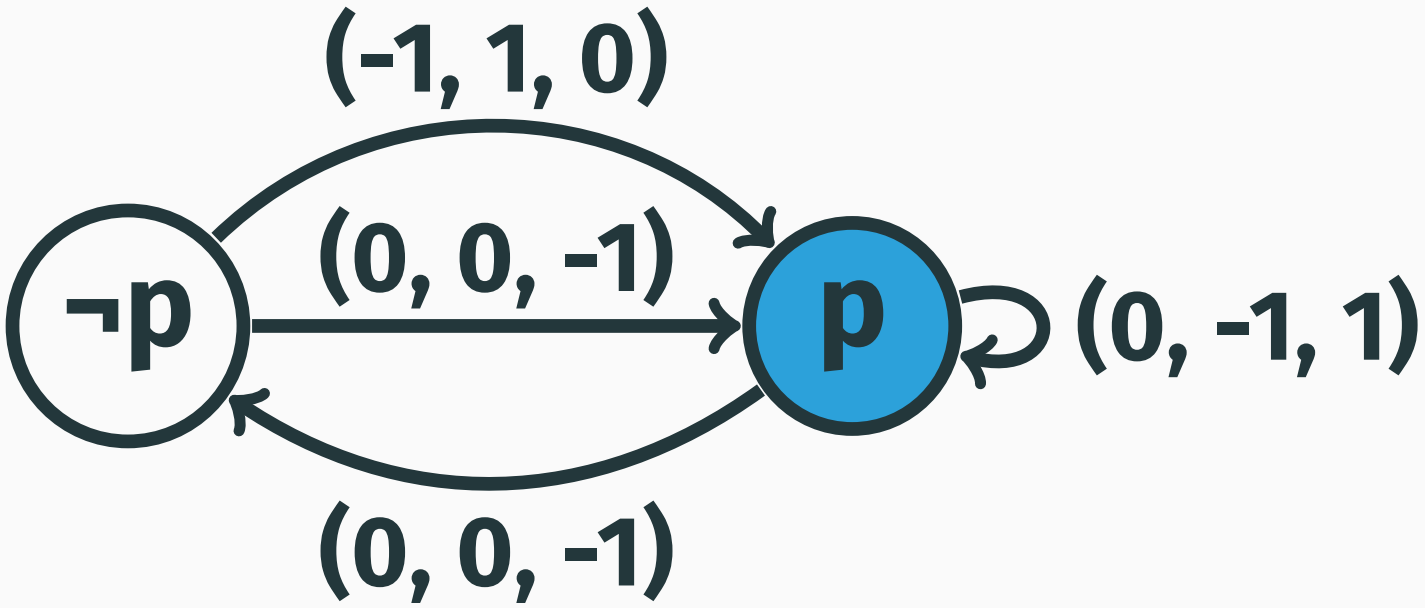

**¬p(5, 0, 0)**

# Systèmes d'addition de vecteurs



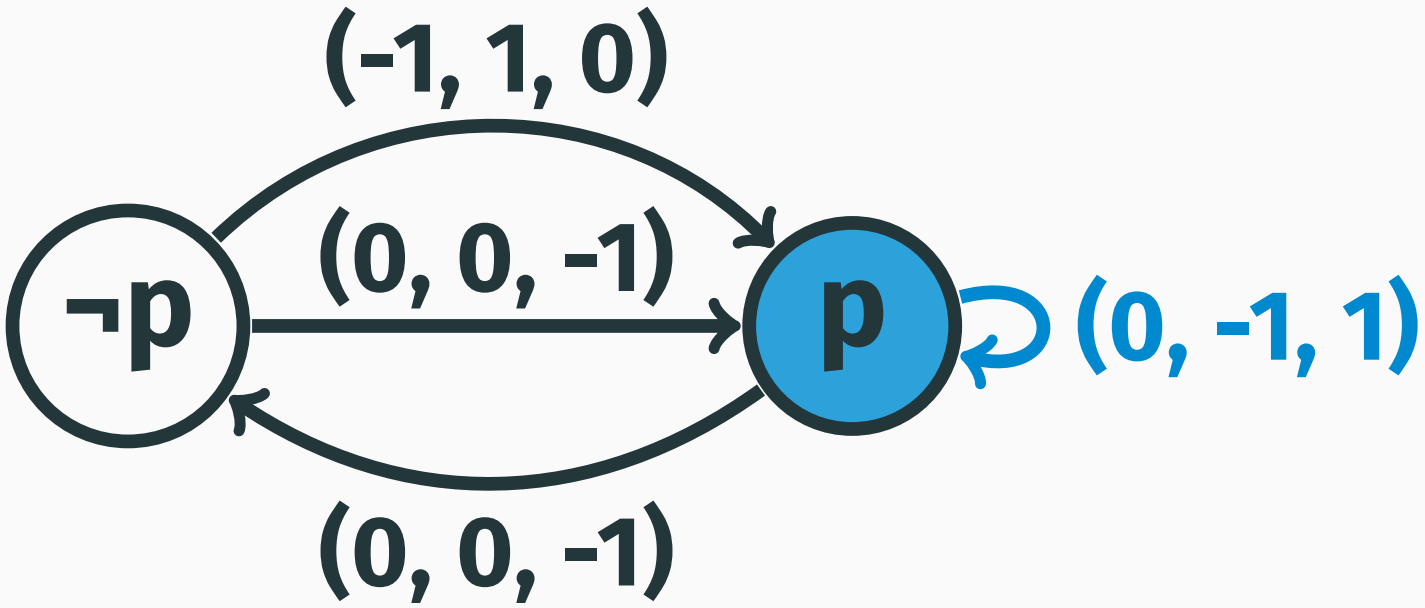

$$\neg p(5, 0, 0)$$

# Systèmes d'addition de vecteurs



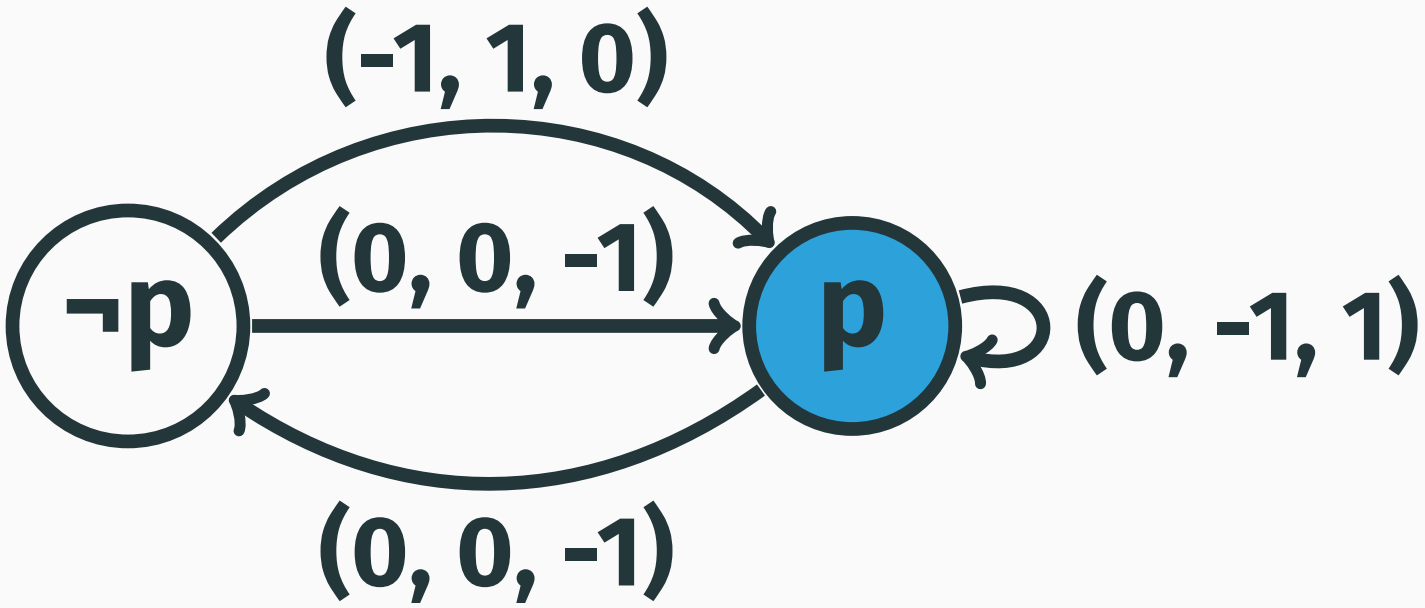

**p(4, 1, 0)**

# Systèmes d'addition de vecteurs



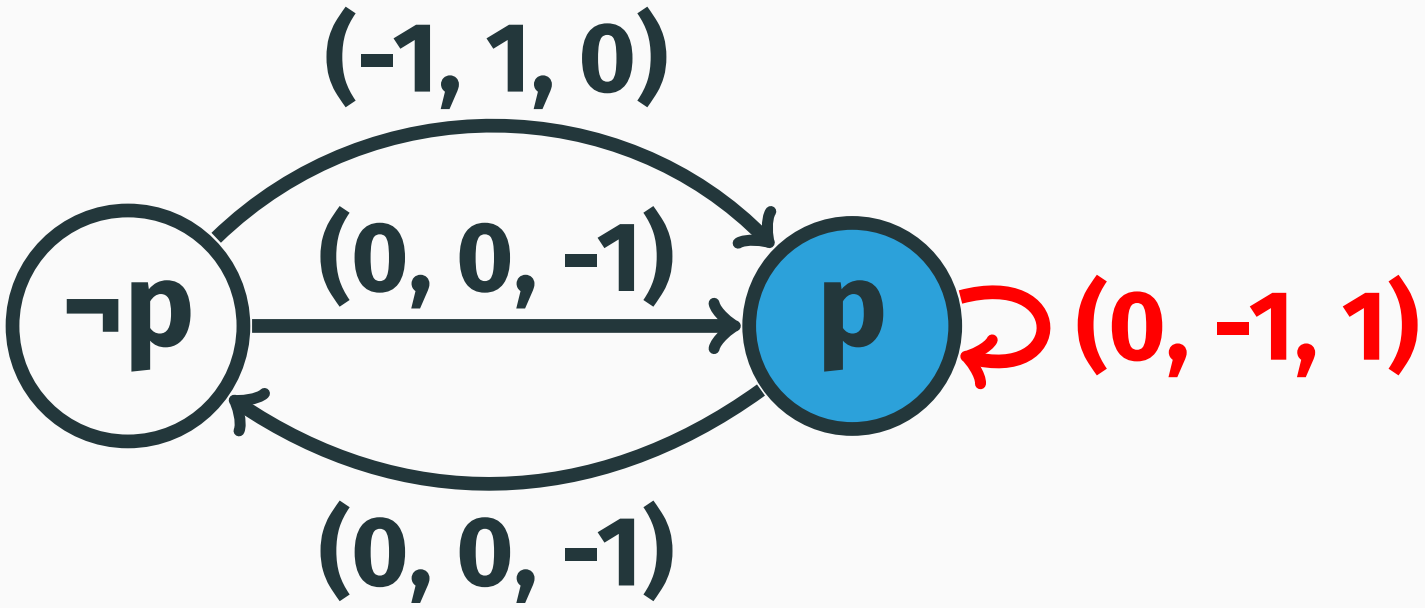

$$p(4, 1, 0)$$

# Systèmes d'addition de vecteurs



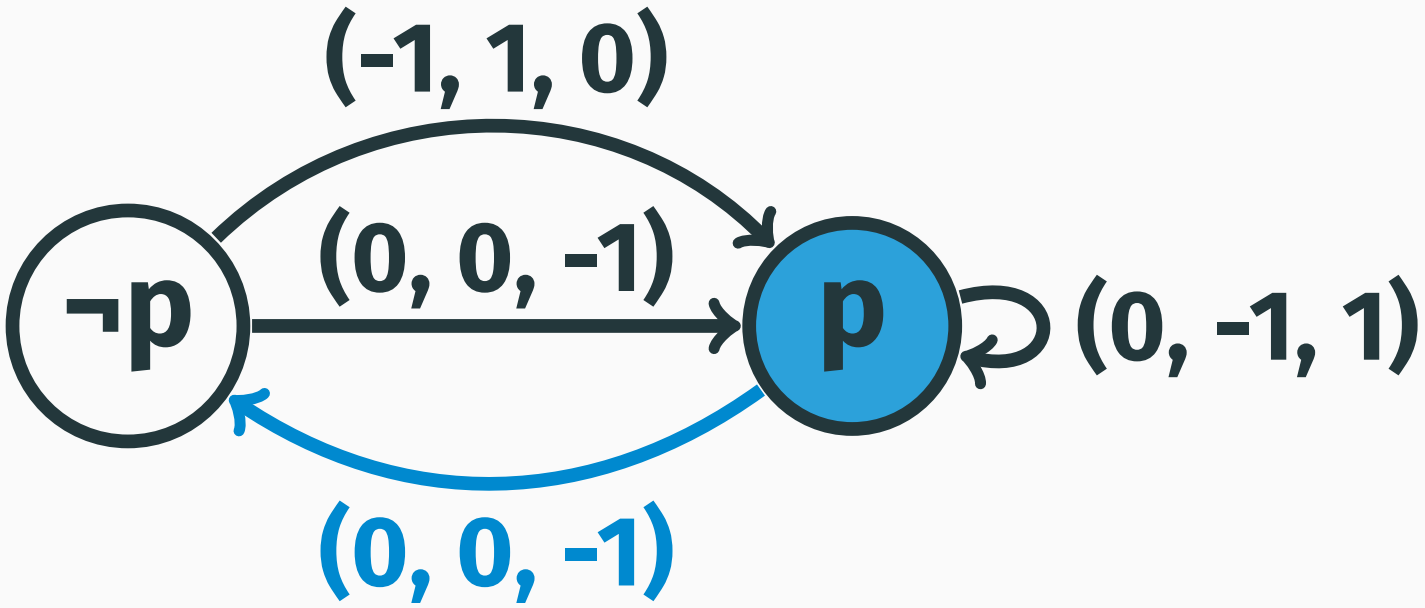

**p(4, 0, 1)**

# Systèmes d'addition de vecteurs



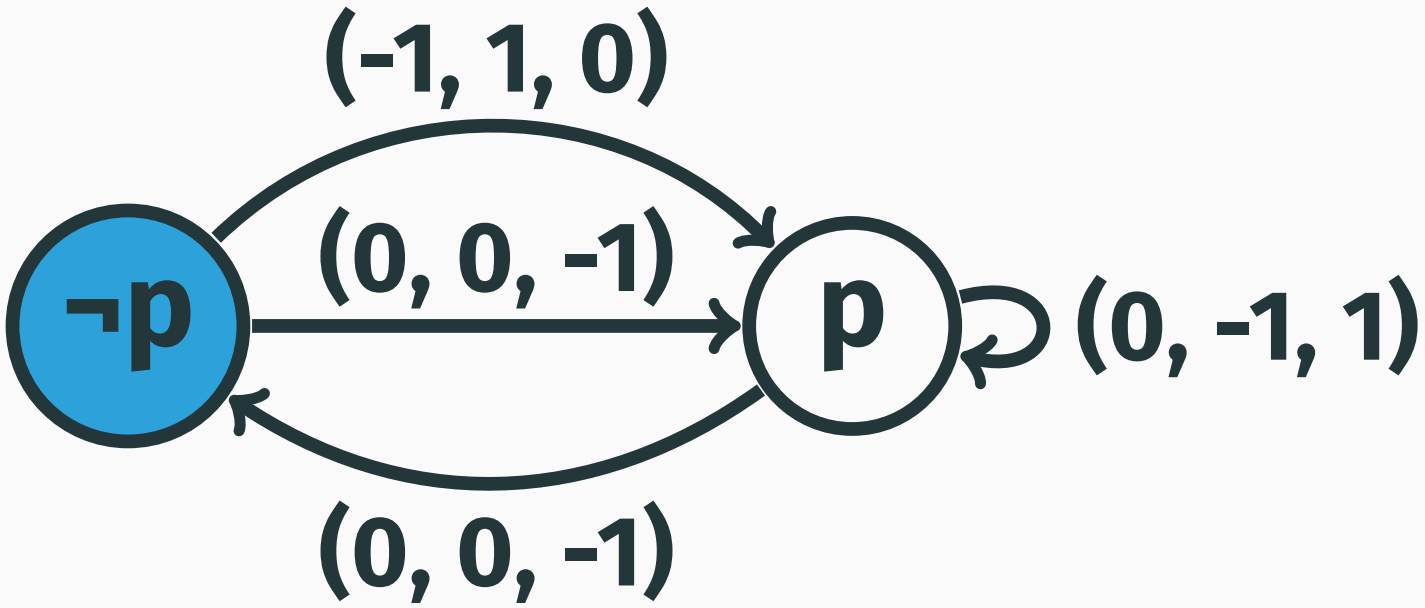

$$p(4, 0, 1)$$

# Systèmes d'addition de vecteurs



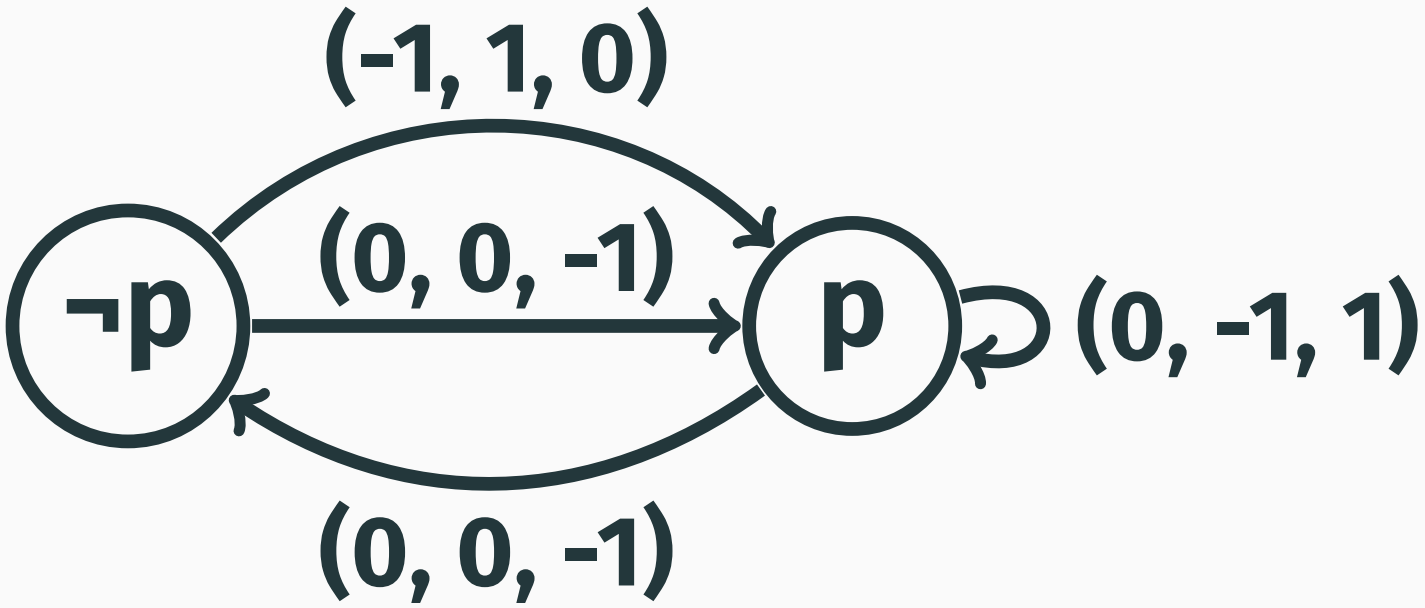

$$p(4, 0, 1)$$

# Systèmes d'addition de vecteurs



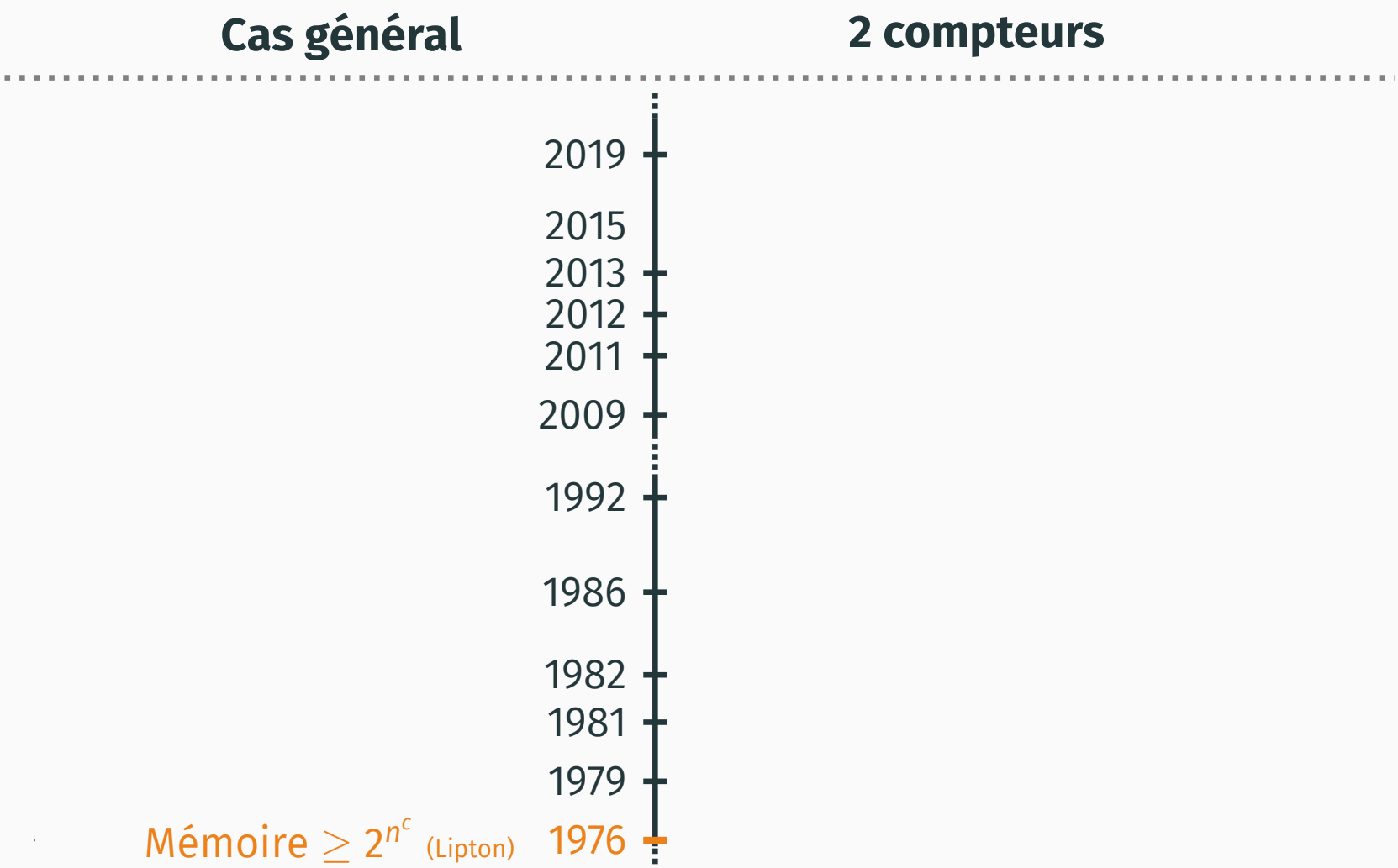

$$\neg p(4, 0, 0)$$

# Systèmes d'addition de vecteurs




$$\neg p(5, 0, 0) \rightsquigarrow p(0, 0, 0)?$$

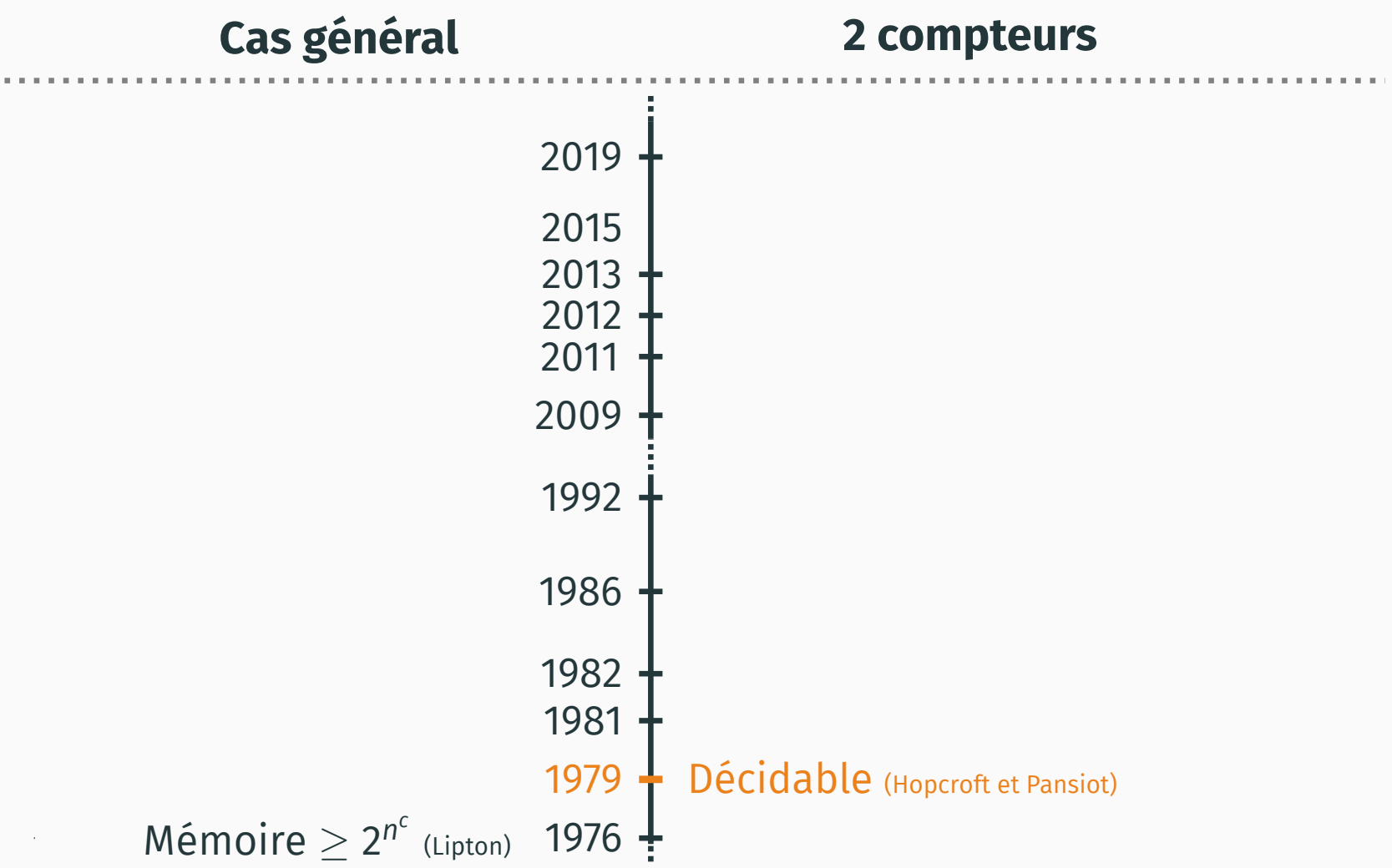
# Problème d'accessibilité




**Entrée:** système et configurations $p(\boldsymbol{u})$ et $q(\boldsymbol{v})$

**Question:** $p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})$?

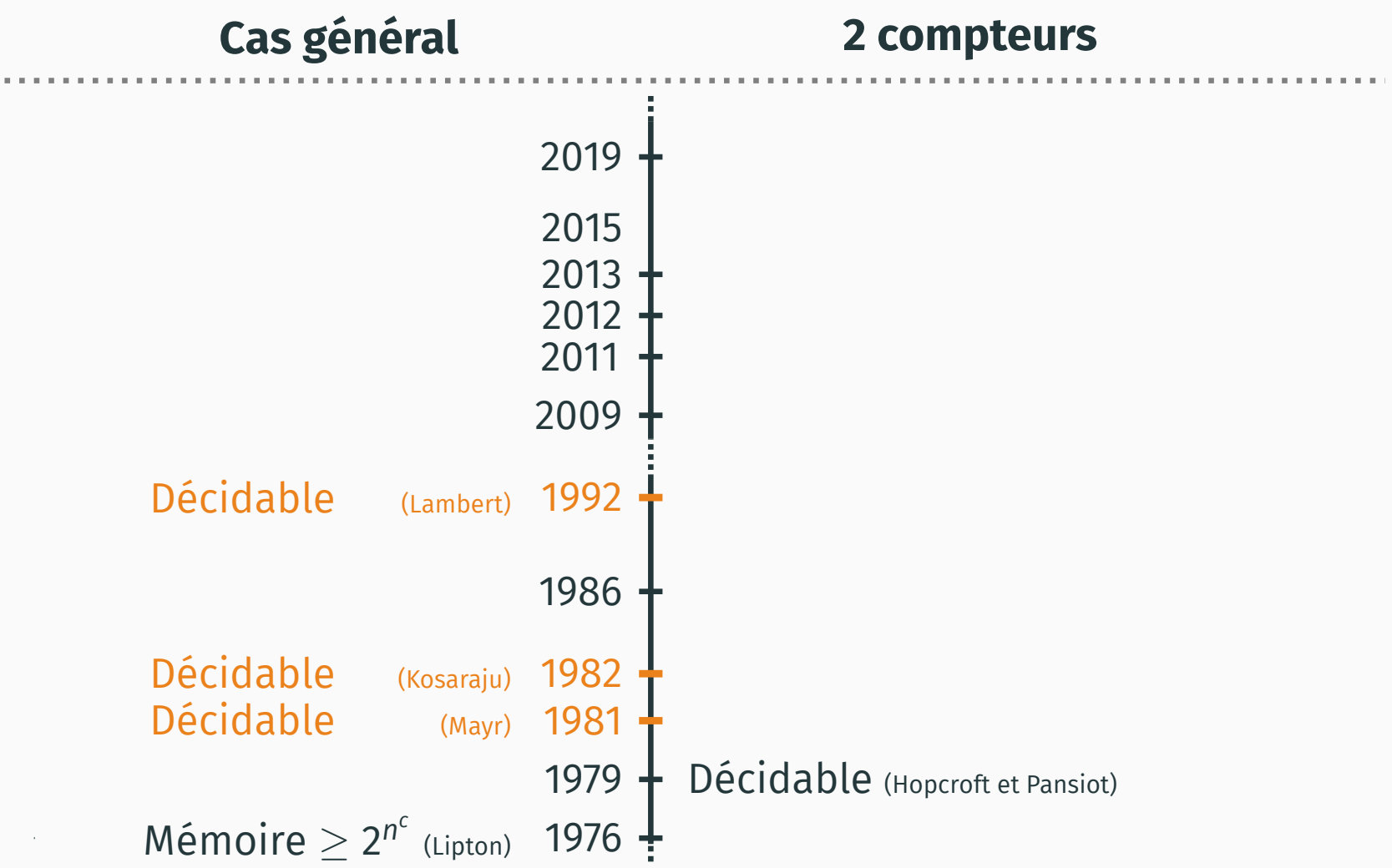
# Problème d'accessibilité




**Entrée:** système et configurations $p(\boldsymbol{u})$ et $q(\boldsymbol{v})$

**Question:** $p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})$?

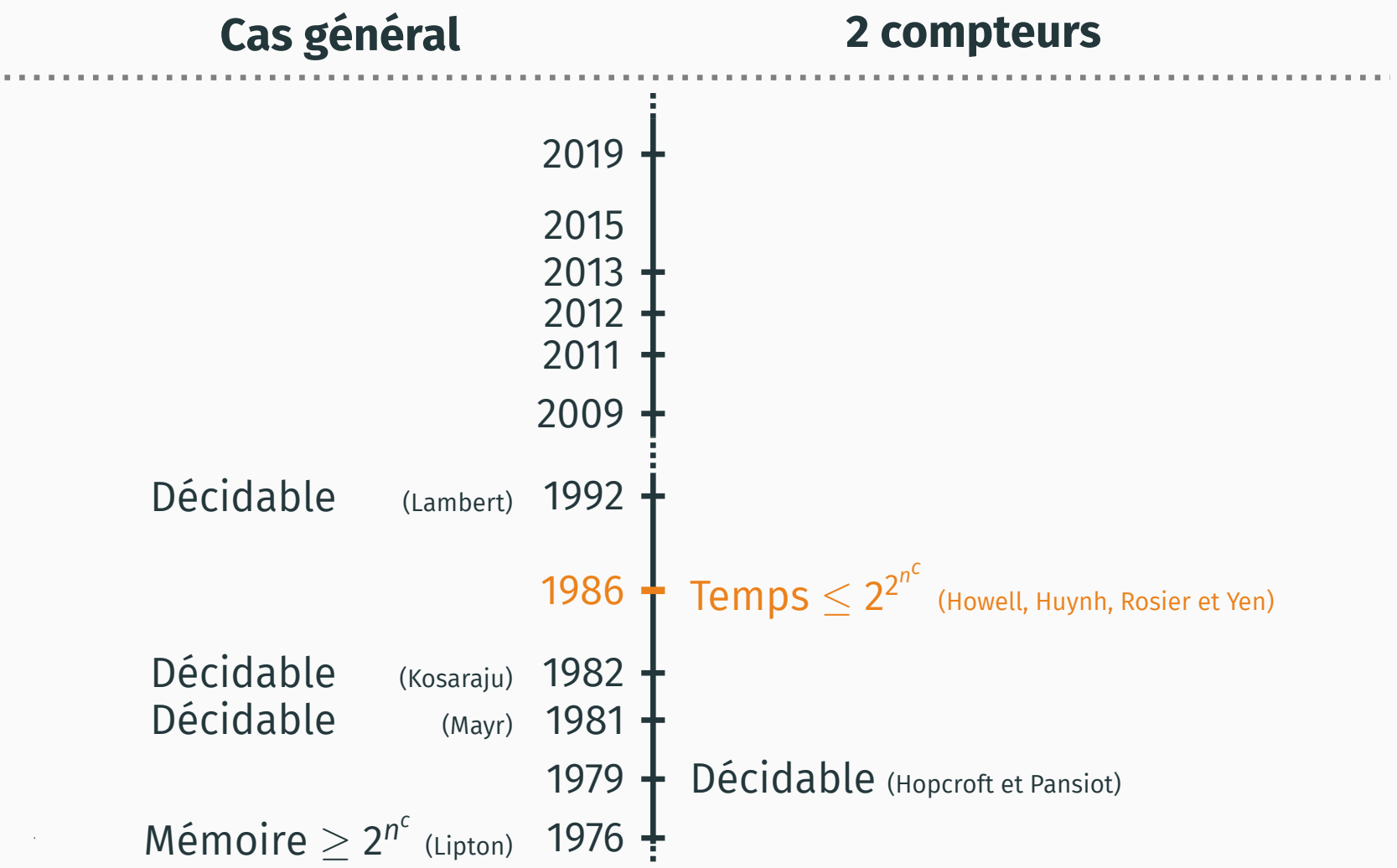
# Problème d'accessibilité




**Entrée:** système et configurations $p(\boldsymbol{u})$ et $q(\boldsymbol{v})$

**Question:** $p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})$?

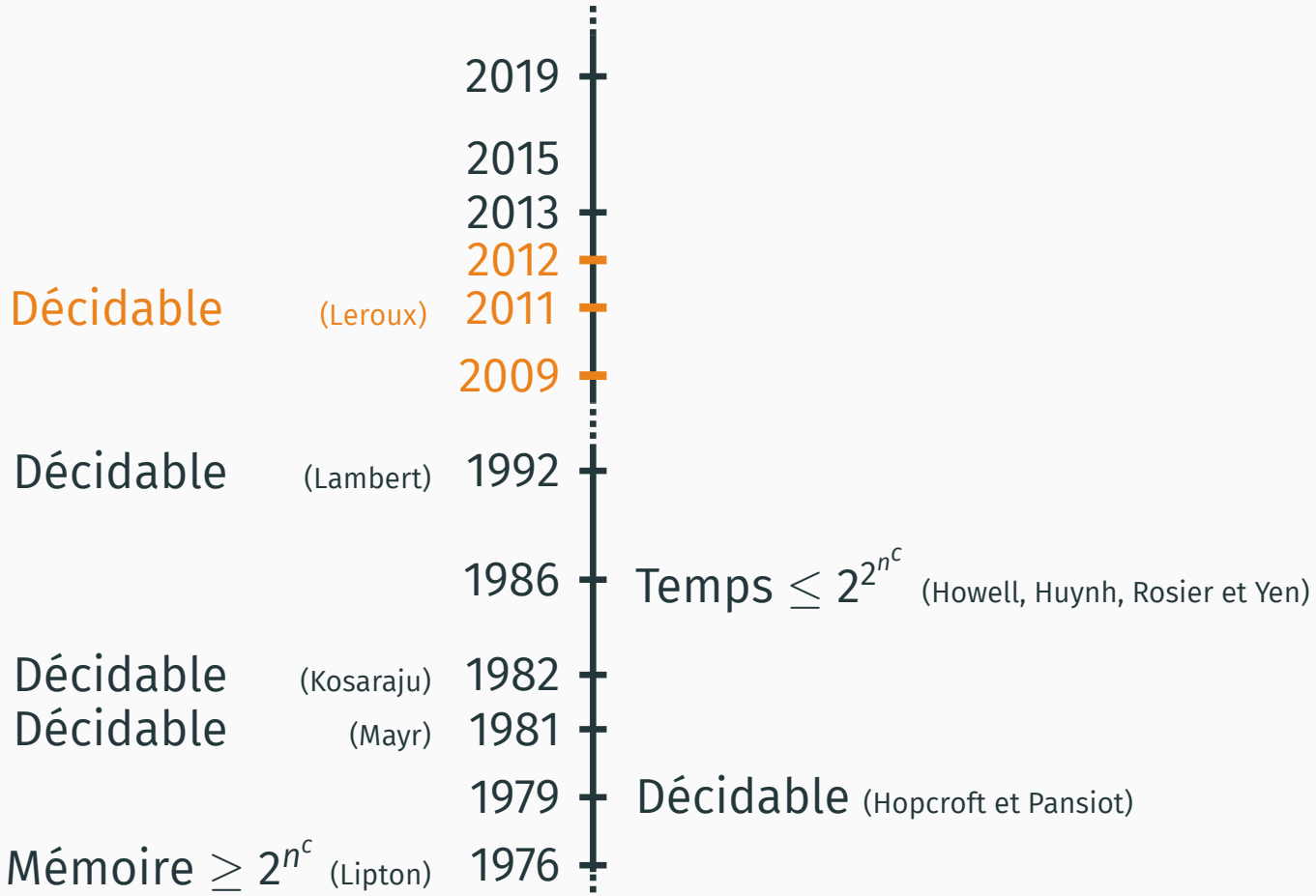
# Problème d'accessibilité




**Entrée:** système et configurations $p(\boldsymbol{u})$ et $q(\boldsymbol{v})$

**Question:** $p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})$?

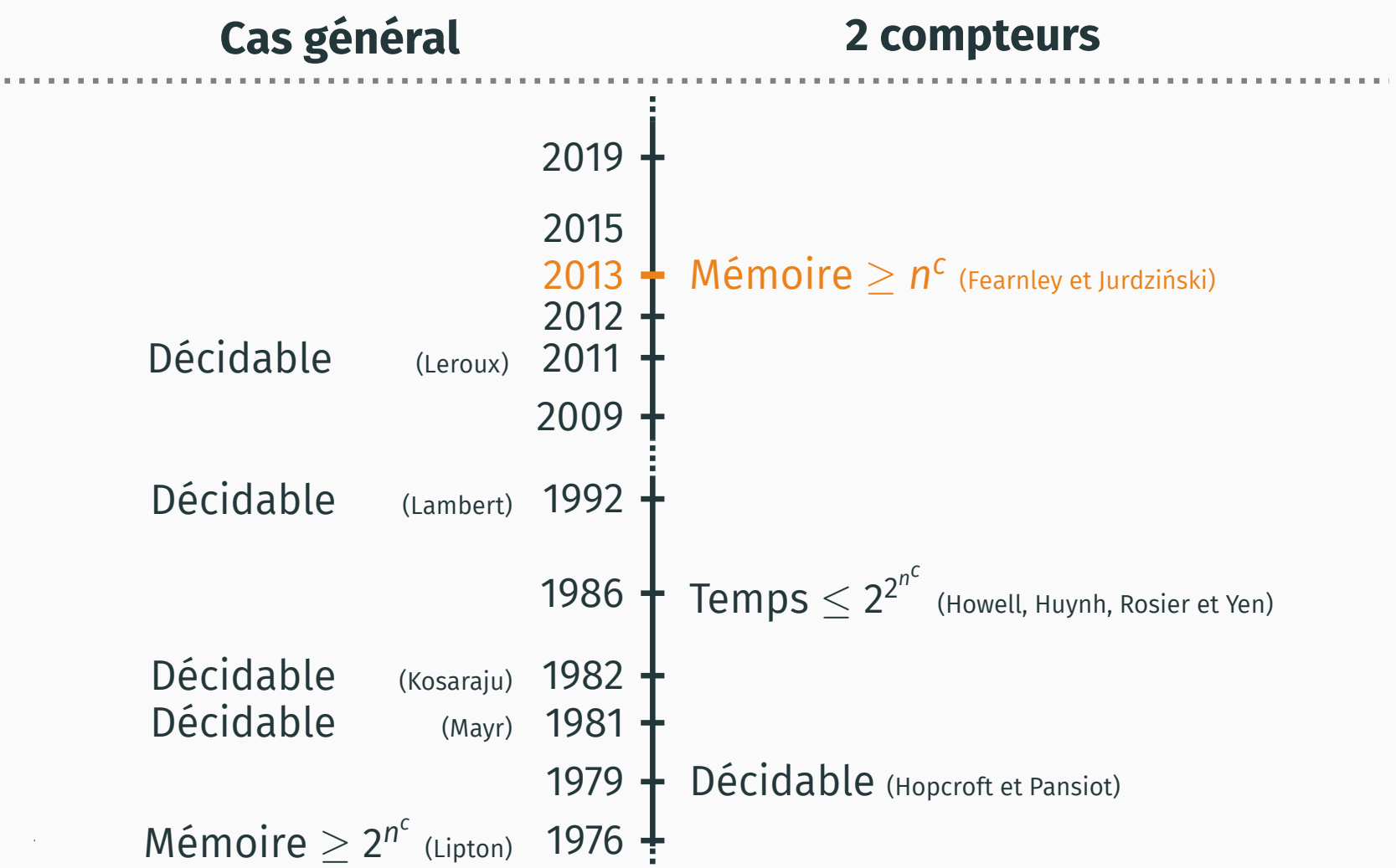
# Problème d'accessibilité




**Entrée:** système et configurations $p(\boldsymbol{u})$ et $q(\boldsymbol{v})$

**Question:** $p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})$?

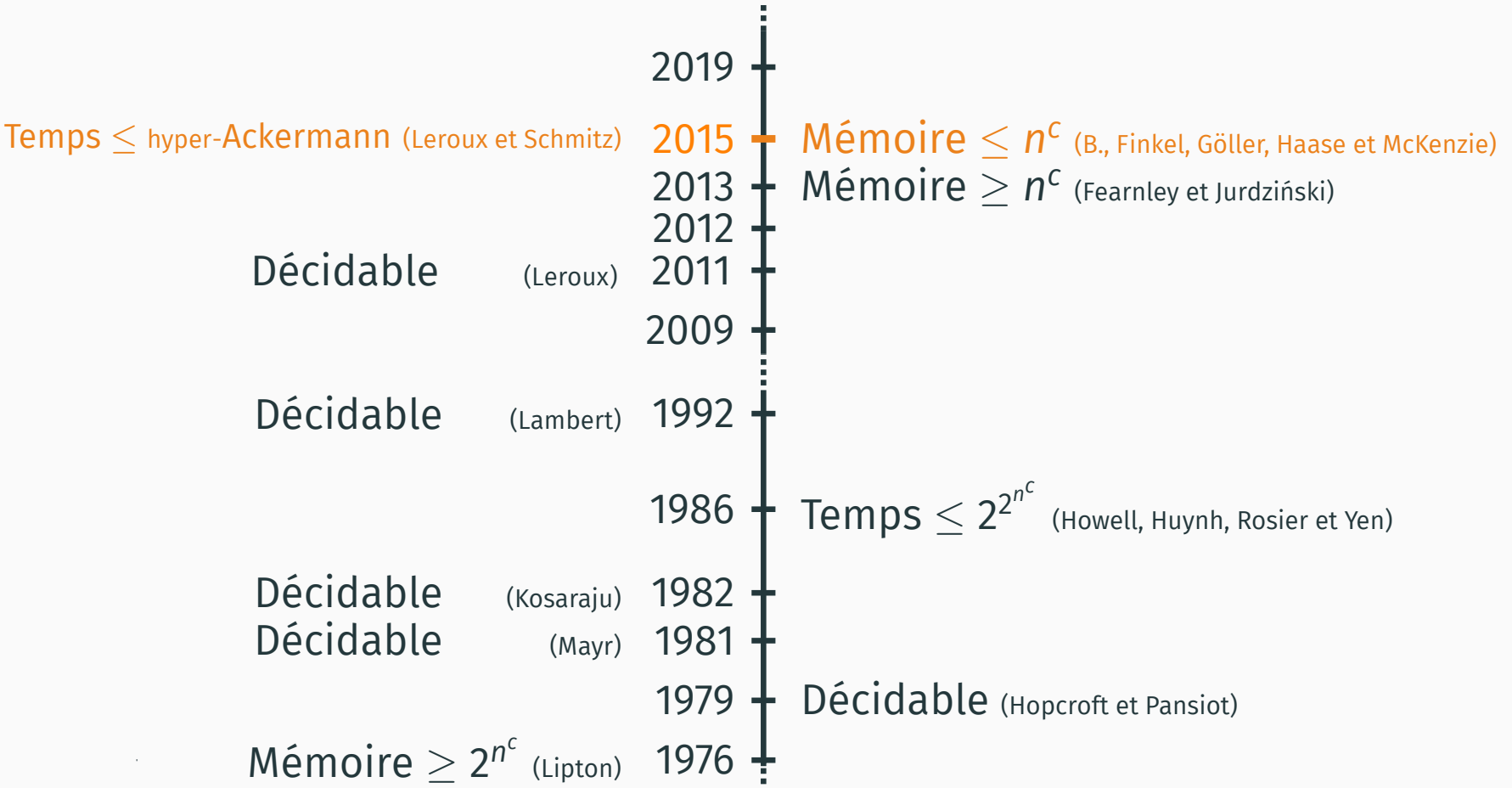
# Problème d'accessibilité




**Entrée:** système et configurations $p(\boldsymbol{u})$ et $q(\boldsymbol{v})$

**Question:** $p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})$?

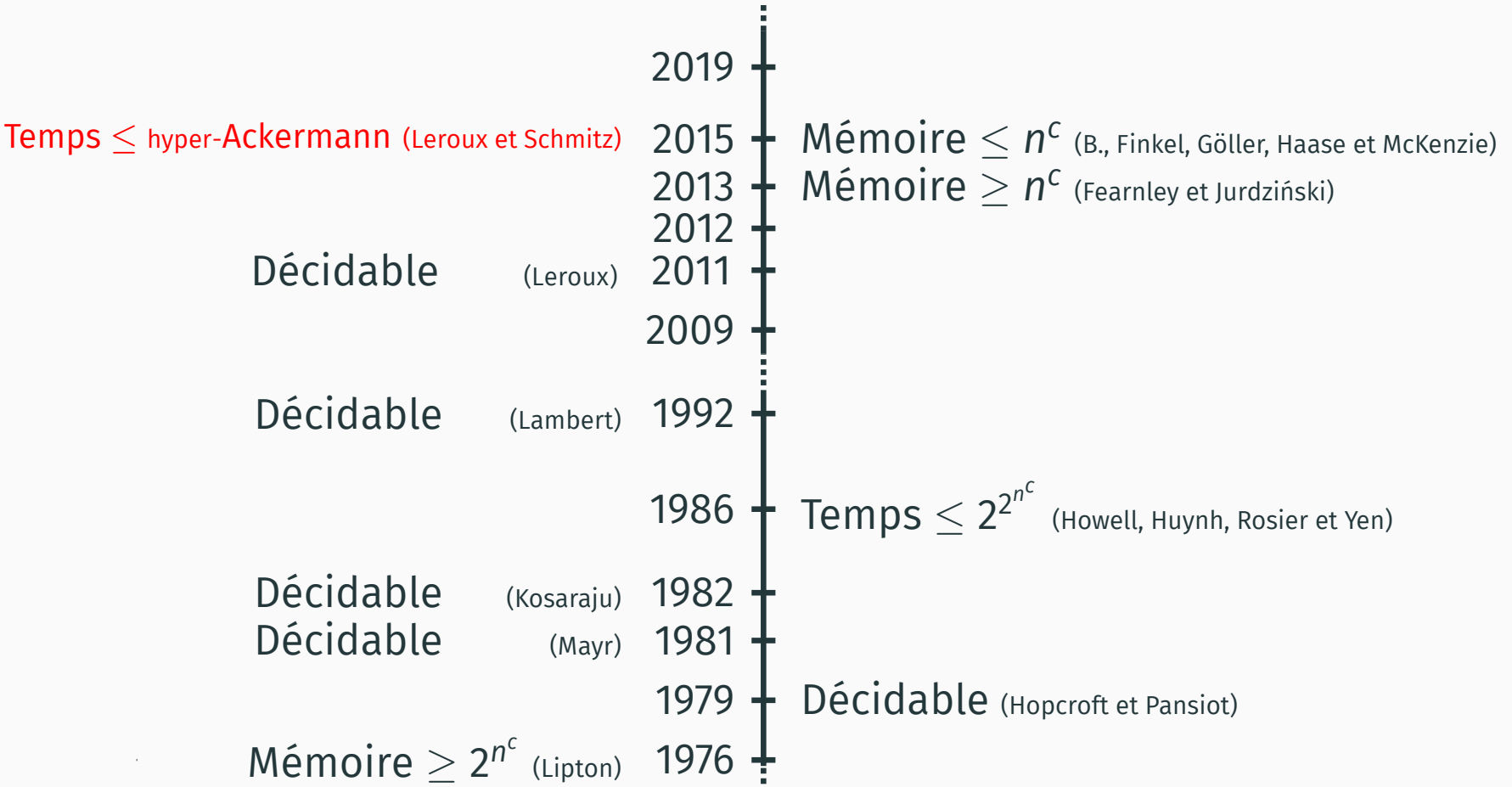
# Problème d'accessibilité




**Entrée:** système et configurations $p(\boldsymbol{u})$ et $q(\boldsymbol{v})$

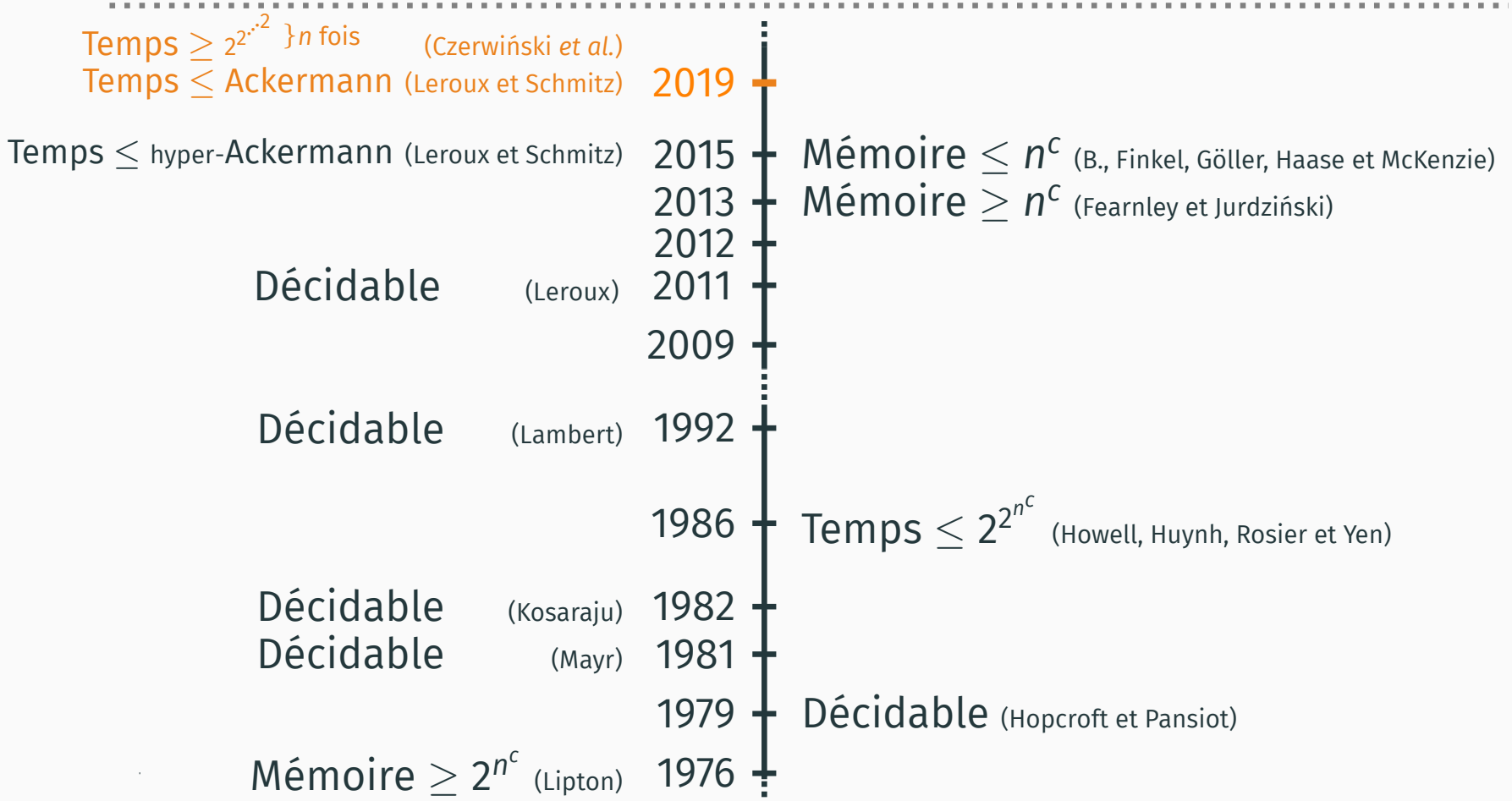
**Question:** $p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})$?

# Problème d'accessibilité

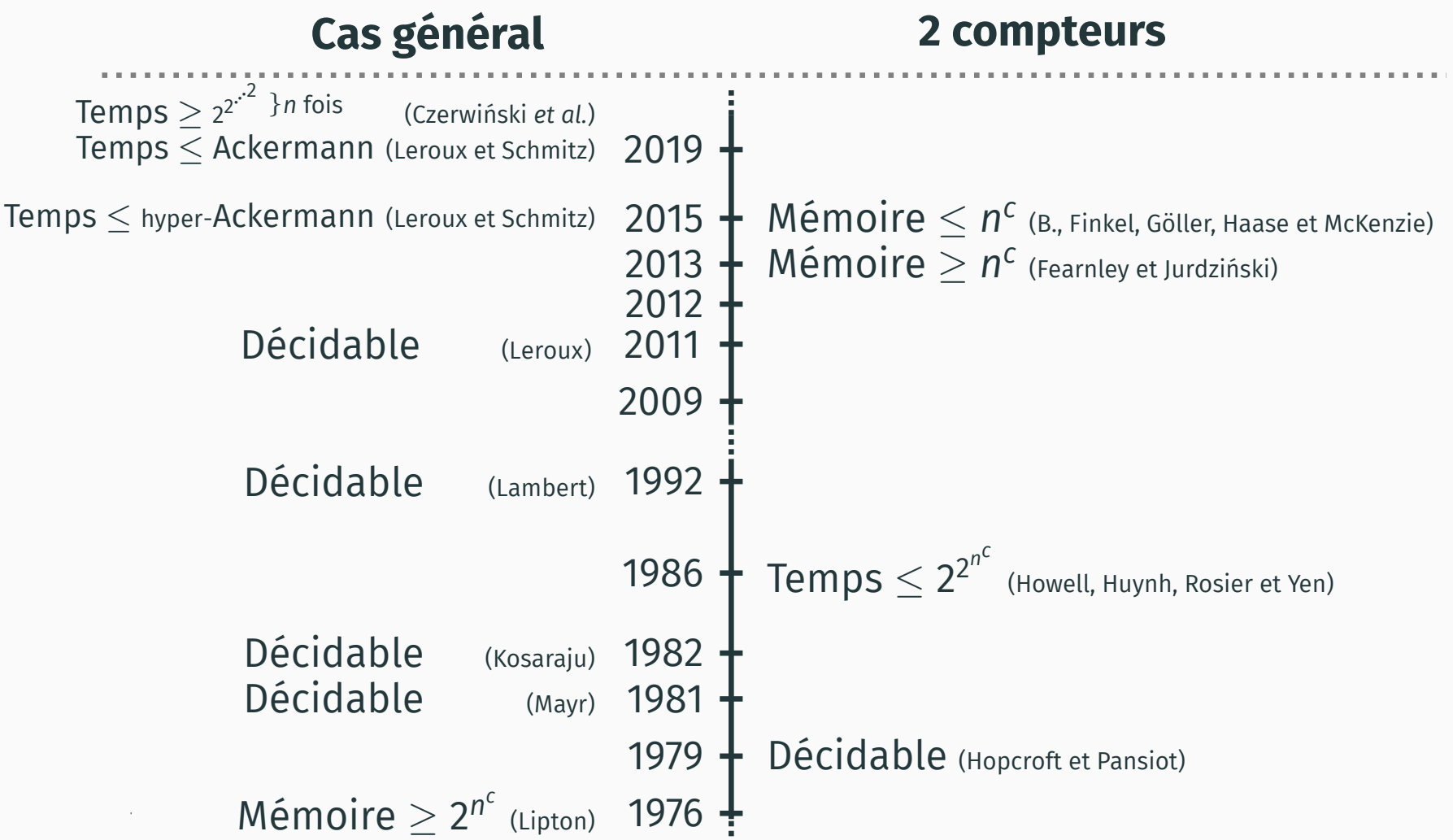



$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0, \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0, \\ A(m-1,A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0. \end{cases}$$

# Problème d'accessibilité



# Problème d'accessibilité




**Défi: surmonter cette complexité** ⚠

# Semilinéarité et arithmétique de Presburger

Si 2 compteurs: $\{\boldsymbol{v} : p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})\}$ est **semilinéaire**

# Semilinéarité et arithmétique de Presburger

Si 2 compteurs: $\{\boldsymbol{v} : p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})\}$ est **semilinéaire**

$$\boldsymbol{b} + \qquad \mathbb{N} \cdot \boldsymbol{x}_1 + \ldots + \mathbb{N} \cdot \boldsymbol{x}_k$$

# Semilinéarité et arithmétique de Presburger

Si 2 compteurs: $\{\boldsymbol{v} : p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})\}$ est **semilinéaire**

$$\boldsymbol{b} + \bigcup_{\text{finie}} \mathbb{N} \cdot \boldsymbol{x}_1 + \ldots + \mathbb{N} \cdot \boldsymbol{x}_k$$

# Semilinéarité et arithmétique de Presburger

Si 2 compteurs: $\{\boldsymbol{v} : p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})\}$ est **semilinéaire**

$$(0,1) + \mathbb{N} \cdot (2,2) + \mathbb{N} \cdot (0,1)$$

$$=$$

$$\left\{(x,y) \in \mathbb{N}^2 : (x < y) \wedge (\exists k \in \mathbb{N} : x = k + k)\right\}$$

# Semilinéarité et arithmétique de Presburger

Si 2 compteurs: $\{\boldsymbol{v} : p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})\}$ est **semilinéaire**

$$(0,1) + \mathbb{N} \cdot (2,2) + \mathbb{N} \cdot (0,1)$$

$$=$$

$$\left\{(x,y) \in \mathbb{N}^2 : (x < y) \wedge (\exists k \in \mathbb{N} : x = k + k)\right\}$$

**Arithmétique de Presburger: FO$(\mathbb{N}, +)$**

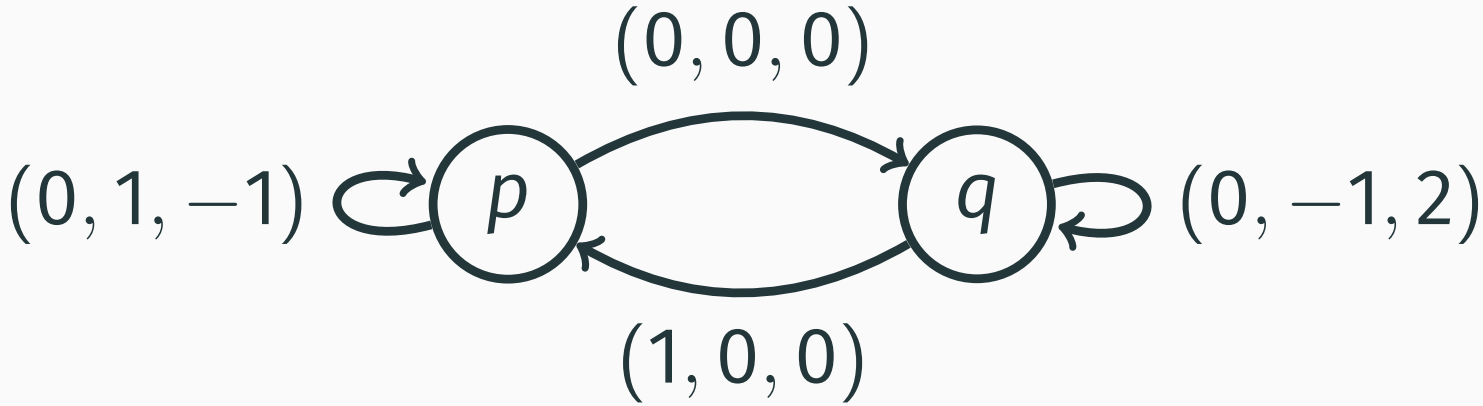
# Semilinéarité et arithmétique de Presburger

Si 2 compteurs: $\{\boldsymbol{v} : p(\boldsymbol{u}) \rightsquigarrow q(\boldsymbol{v})\}$ est **semilinéaire**

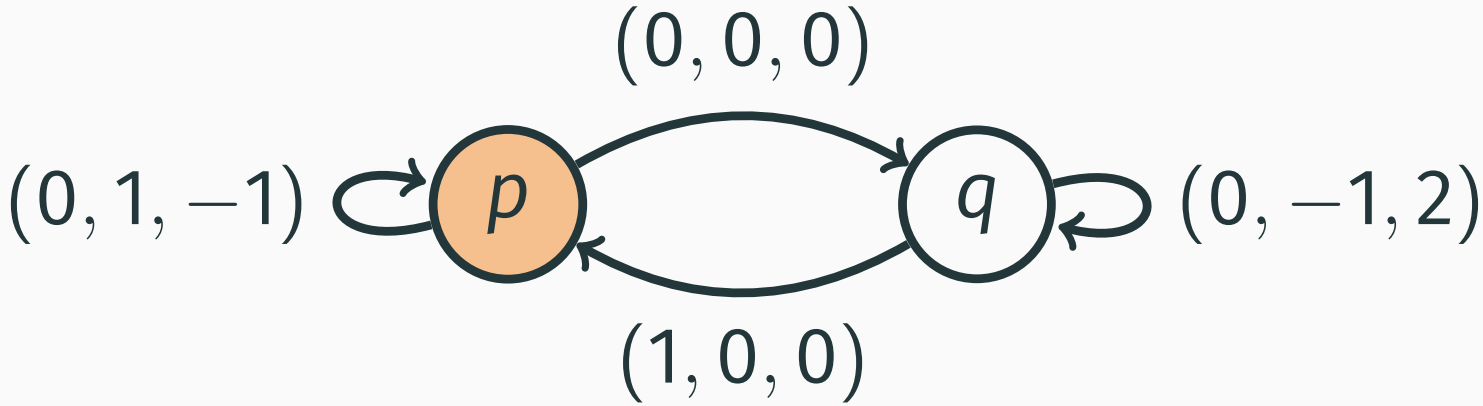
Logique décidable avec solveurs efficaces:
Z3 (Microsoft Research), CVC4, etc.

**Arithmétique de Presburger: FO**$(\mathbb{N}, +)$

Mais, pas semilinéaire pour $\geq 3$ compteurs:

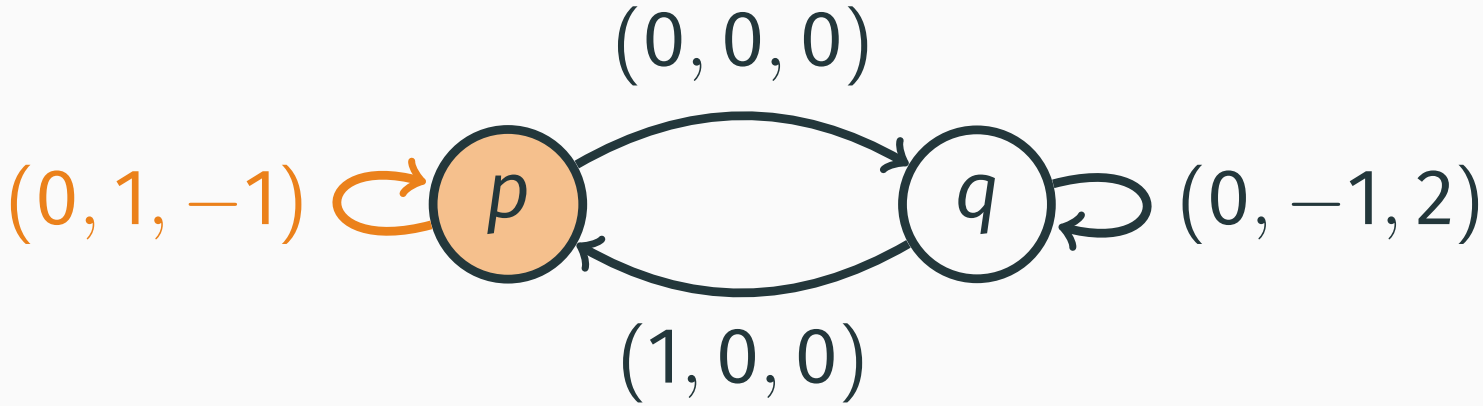


Mais, pas semilinéaire pour $\geq 3$ compteurs:



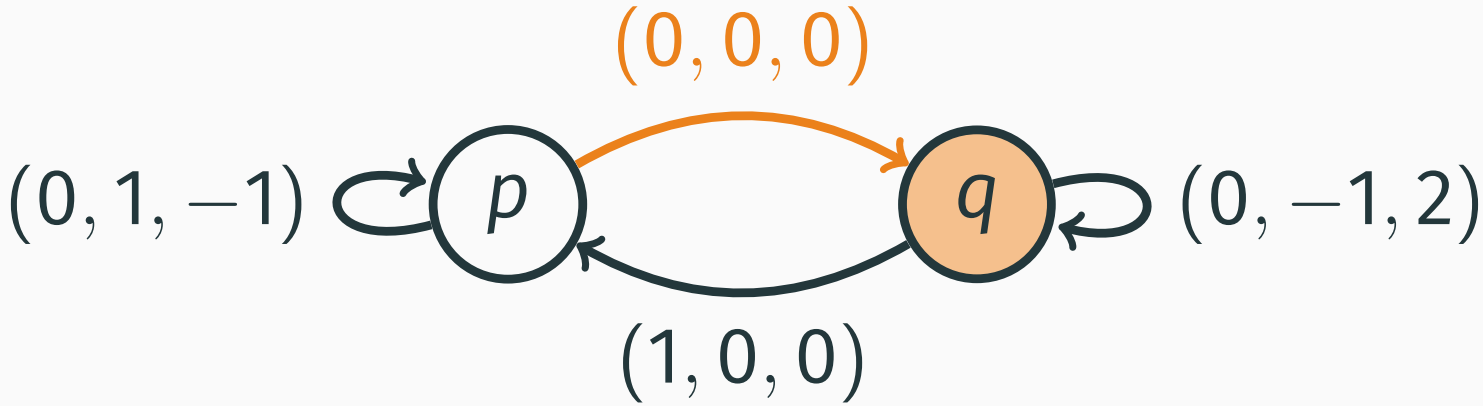

$$p(0,0,1)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



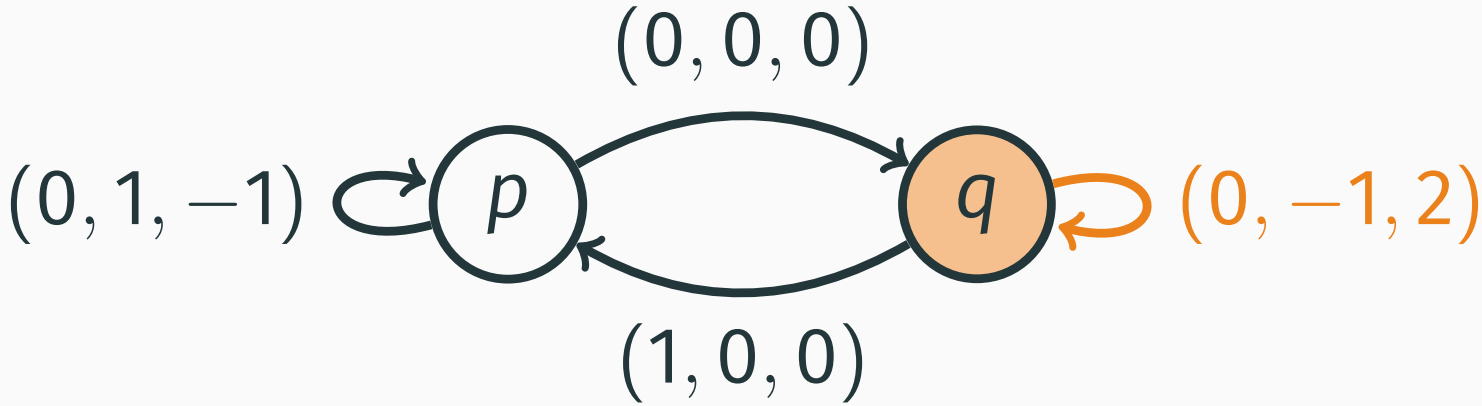

$$p(0,1,0)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



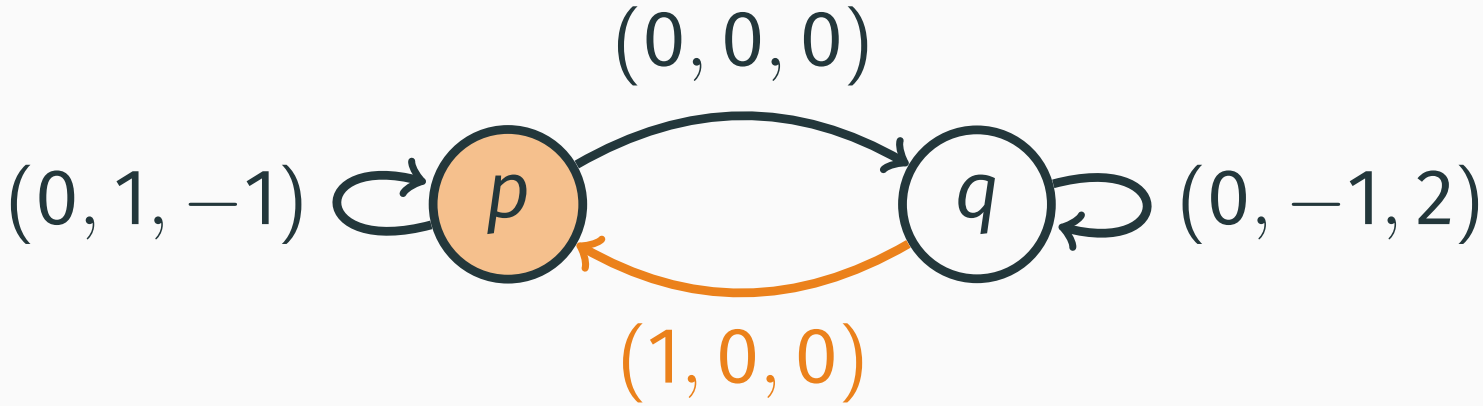

$$q(0,1,0)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



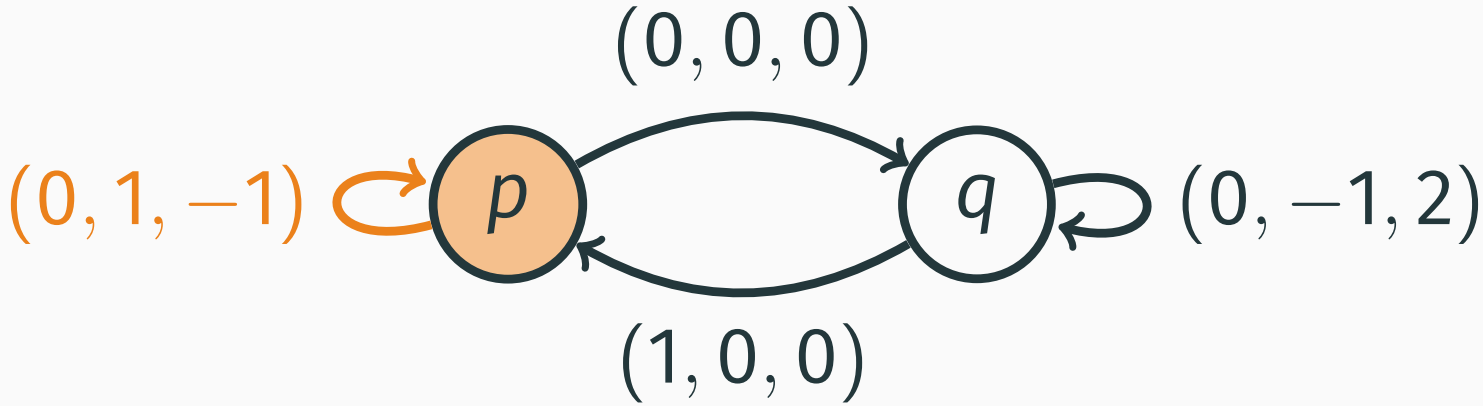

$$q(0,0,2)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



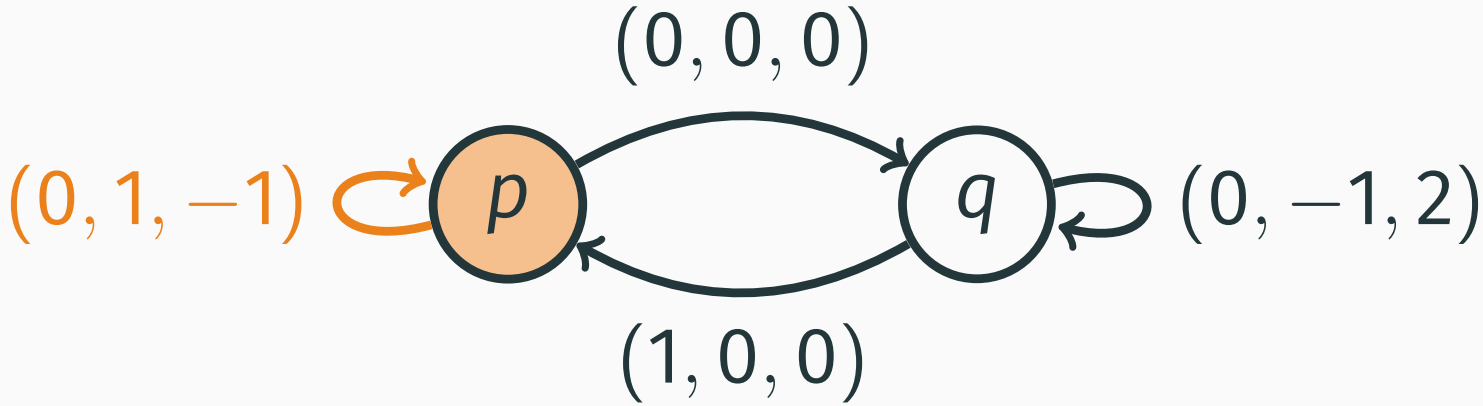

$$p(1,0,2)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



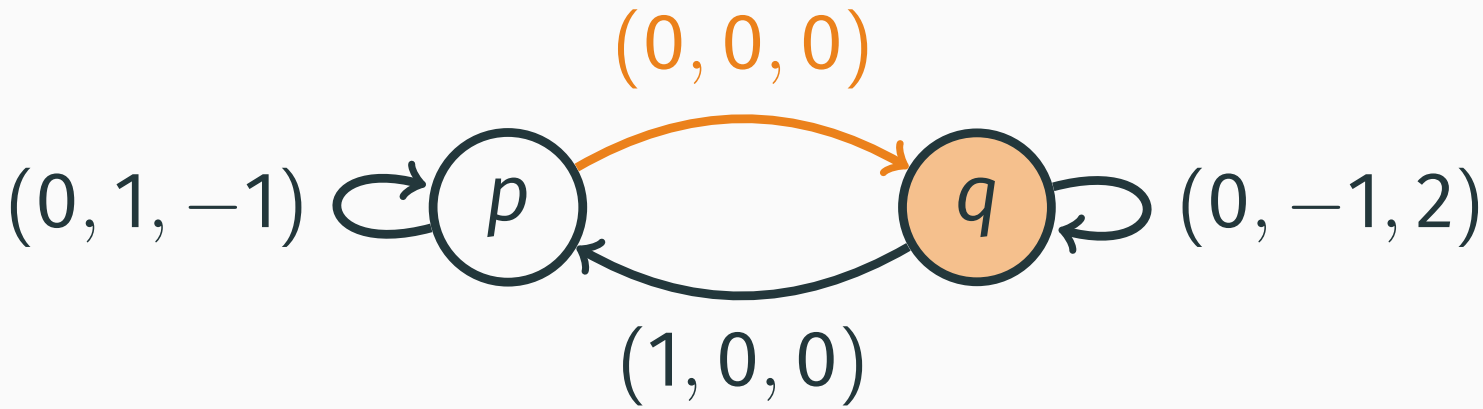

$$p(1,1,1)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



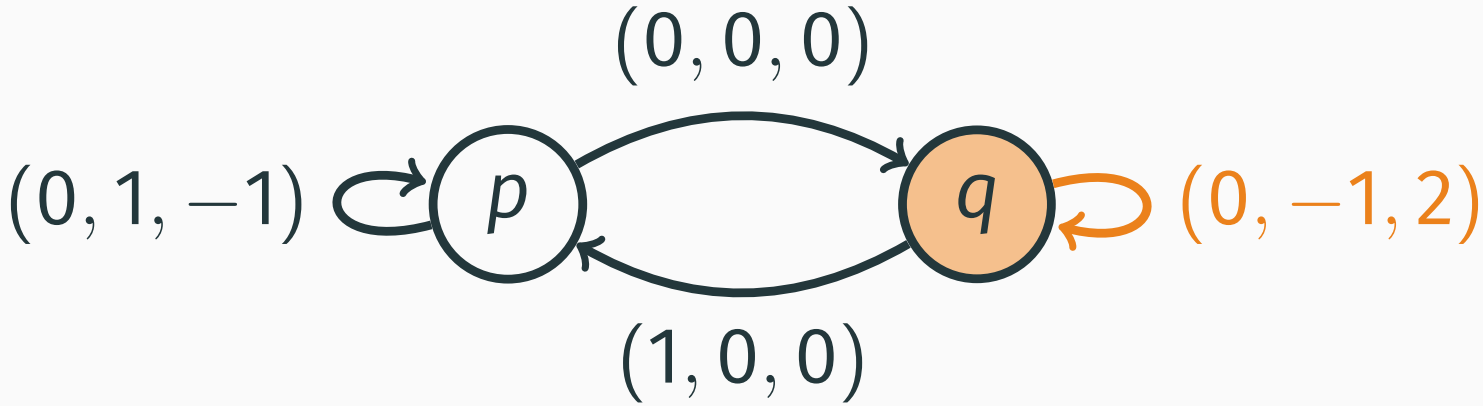

$$p(1,2,0)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



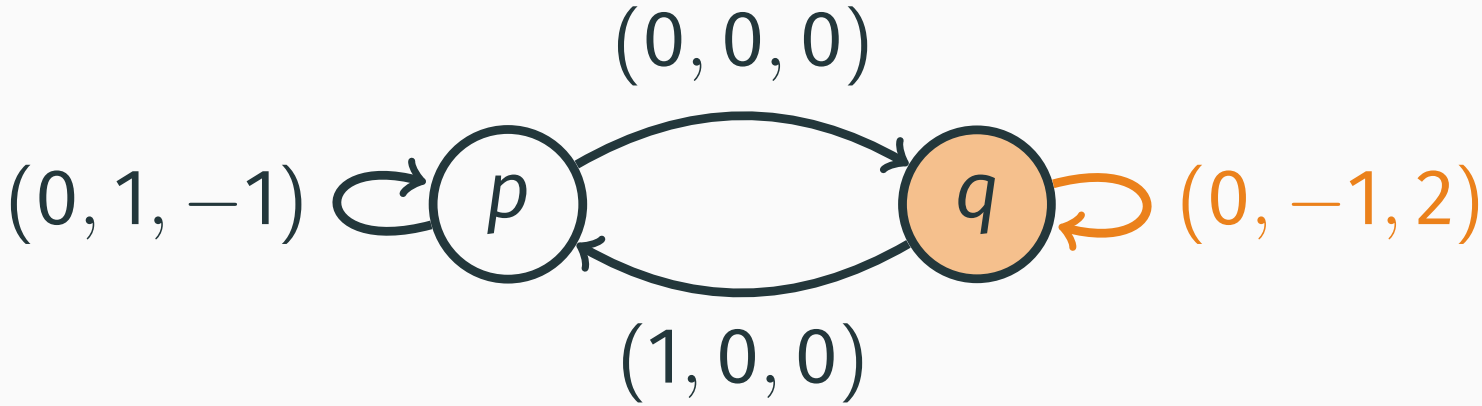

$$q(1,2,0)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



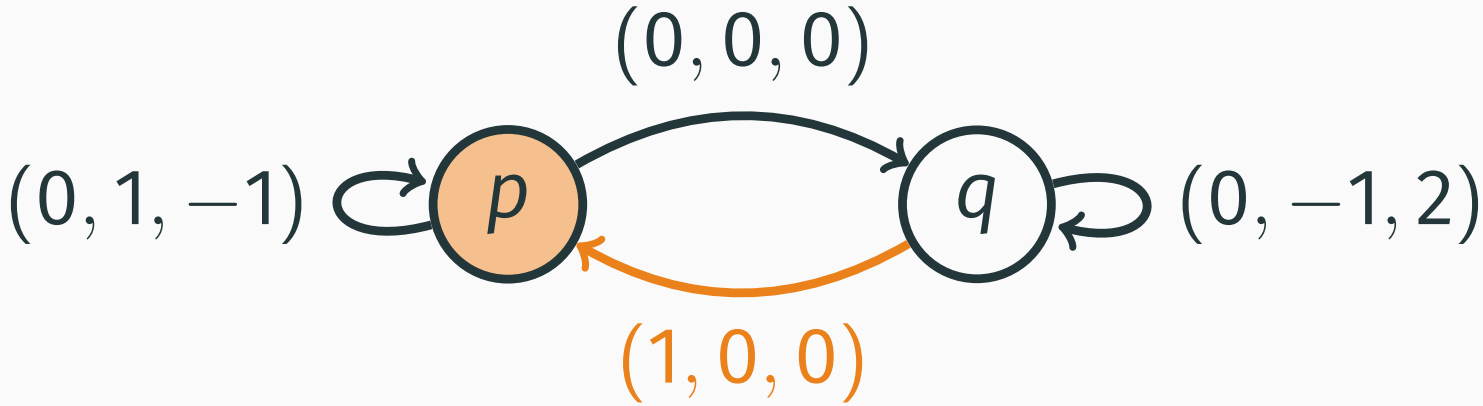

$$q(1,1,2)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



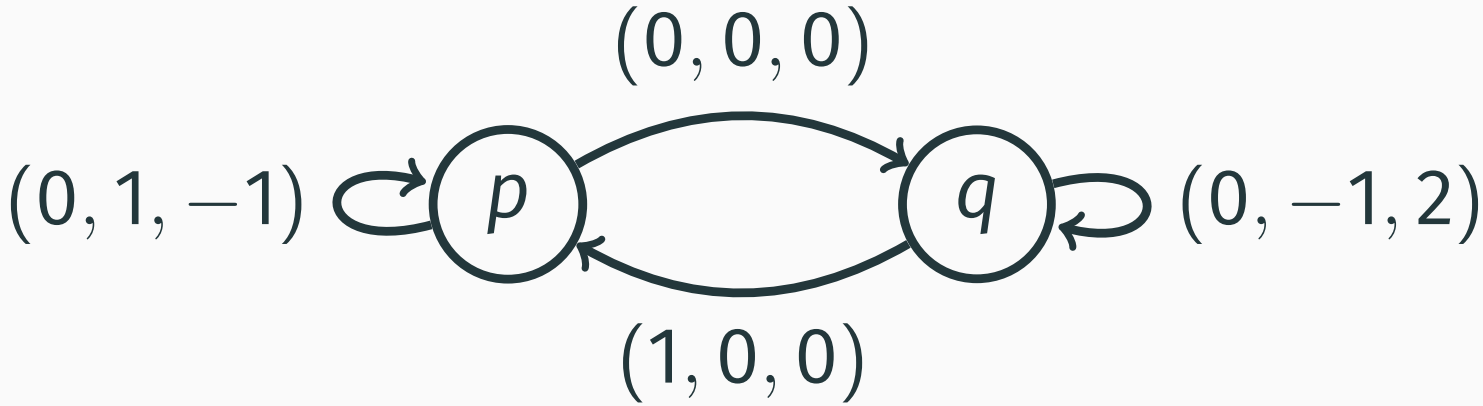

$$q(1,0,4)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:




$$p(2,0,4)$$

Mais, pas semilinéaire pour $\geq 3$ compteurs:



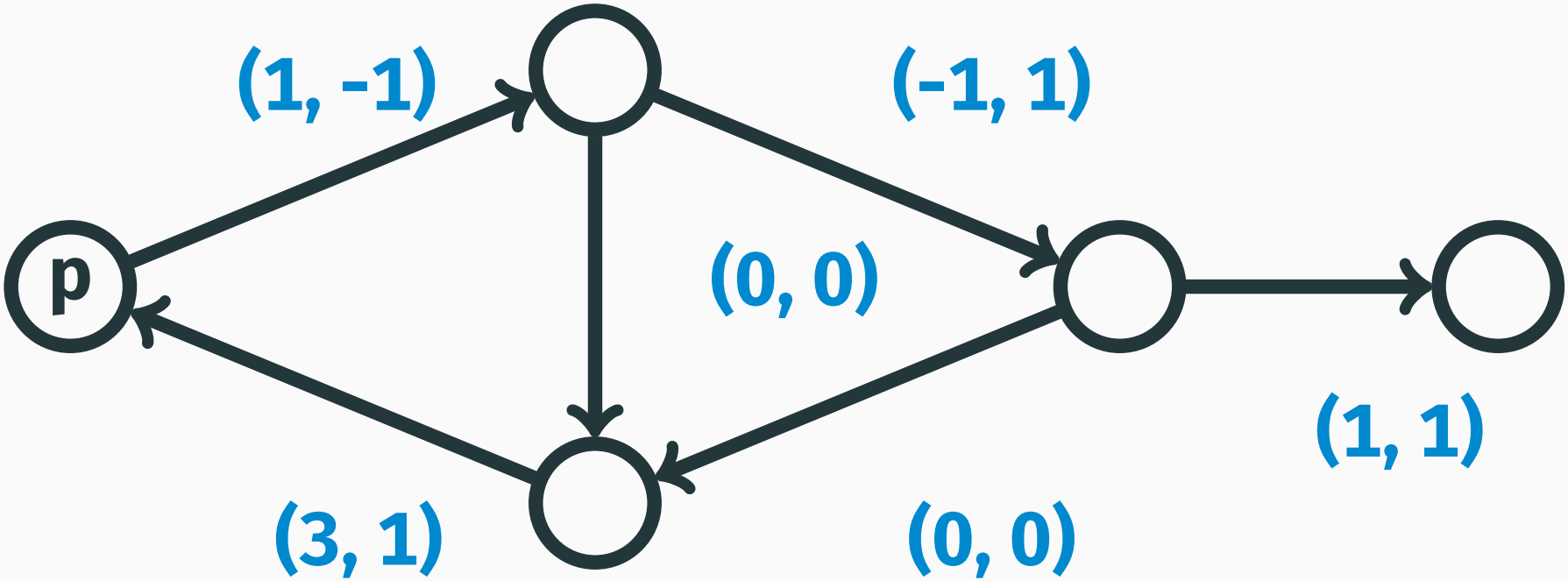

$$p(0,0,1) \rightsquigarrow p(x,y,z) \iff 0 < y+z \leq 2^x$$

**Simplification du problème:**

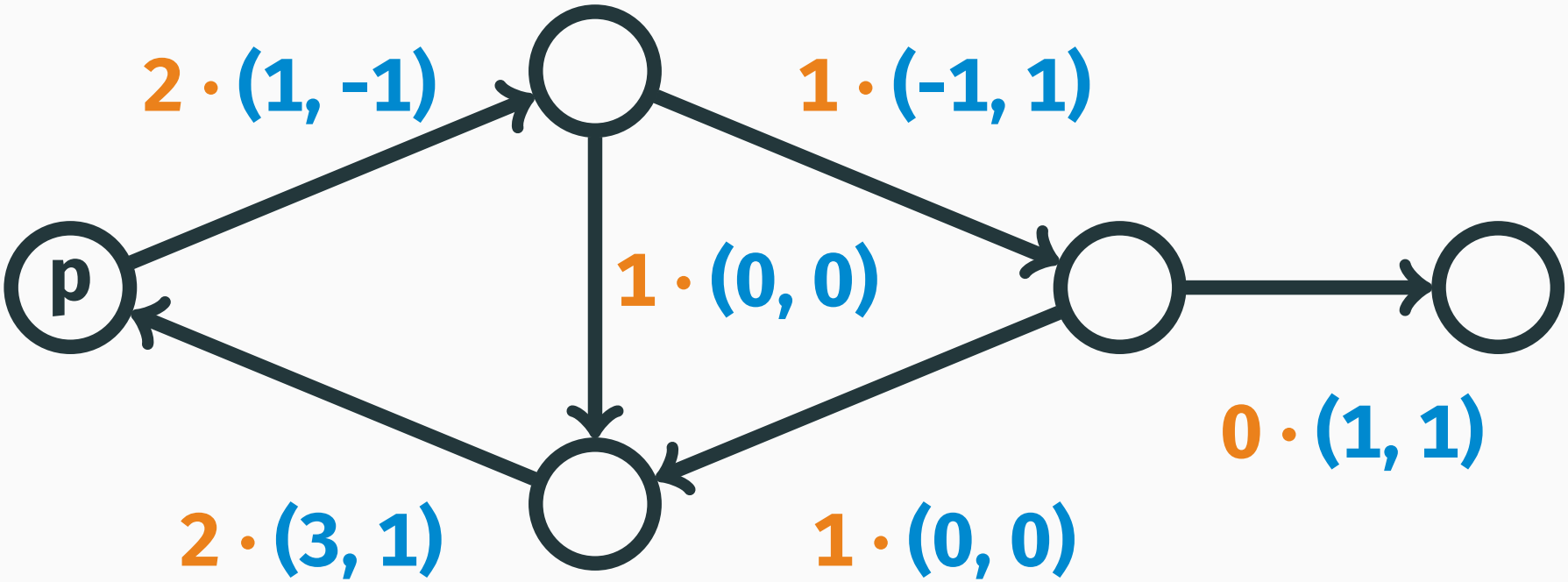
on permet de chuter sous zéro

# Relaxations

$$p(\boldsymbol{u}) \rightsquigarrow_{\mathbb{Z}} p(\boldsymbol{v})?$$

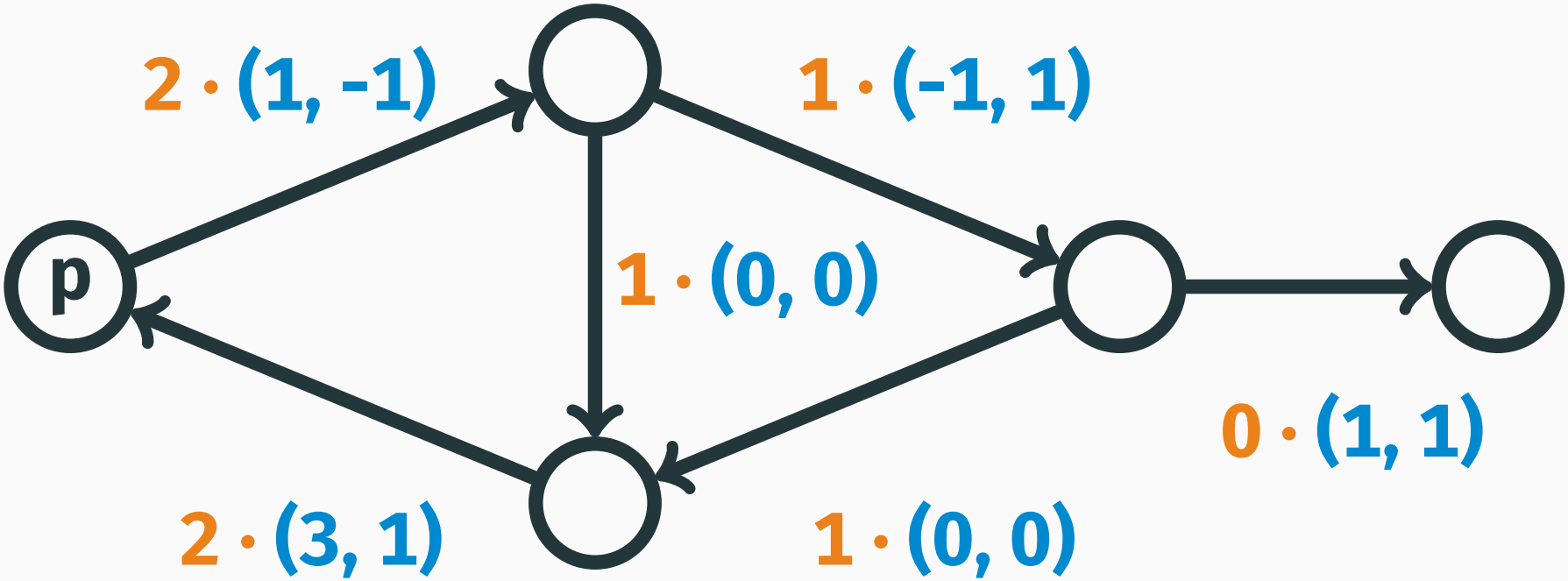


# Deviner nombre d'occurrences

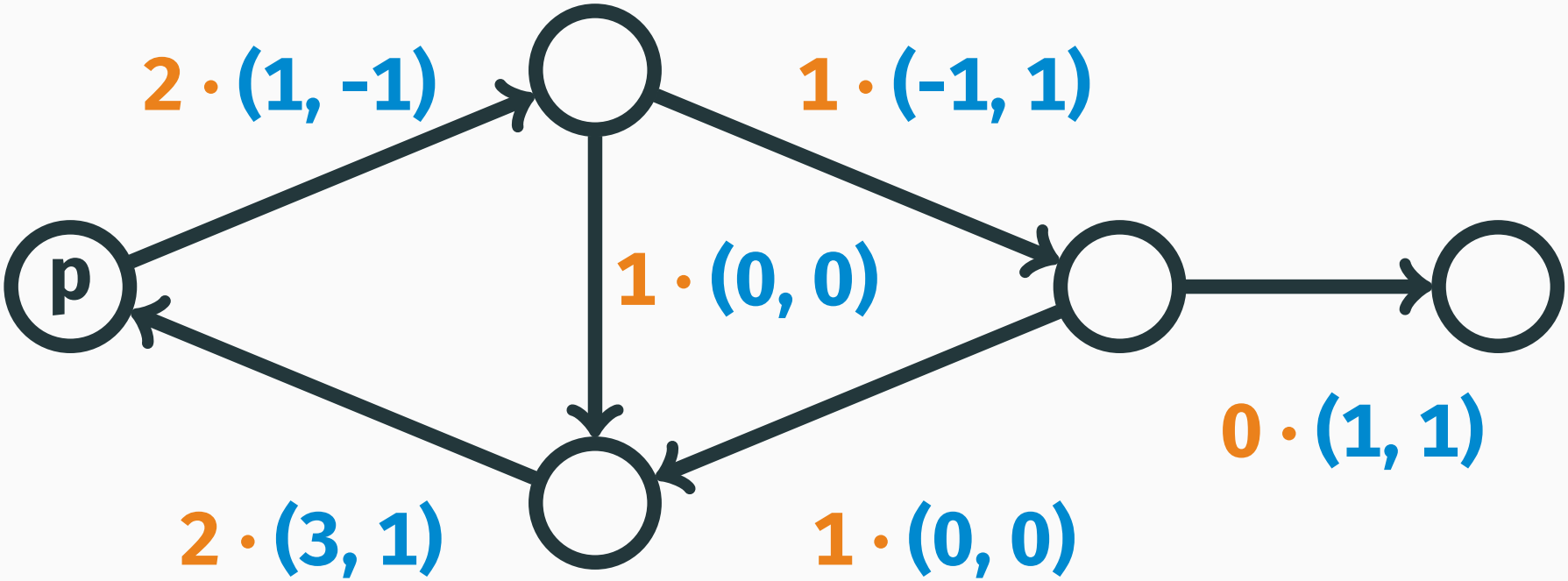


# Relaxations

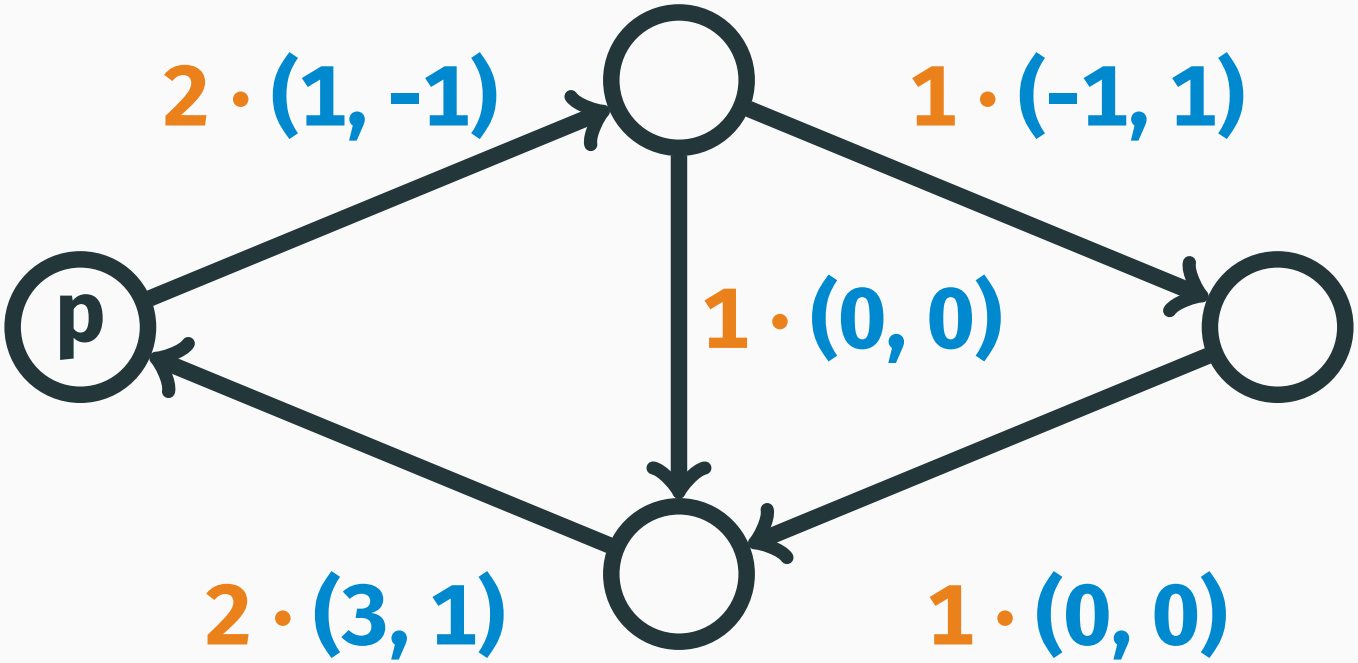
$$\boldsymbol{u} + \sum_{t \in T} \boldsymbol{x}_t \cdot \boldsymbol{\Delta}_t = \boldsymbol{v}$$



# G[$x_t$ > 0] fortement connexe

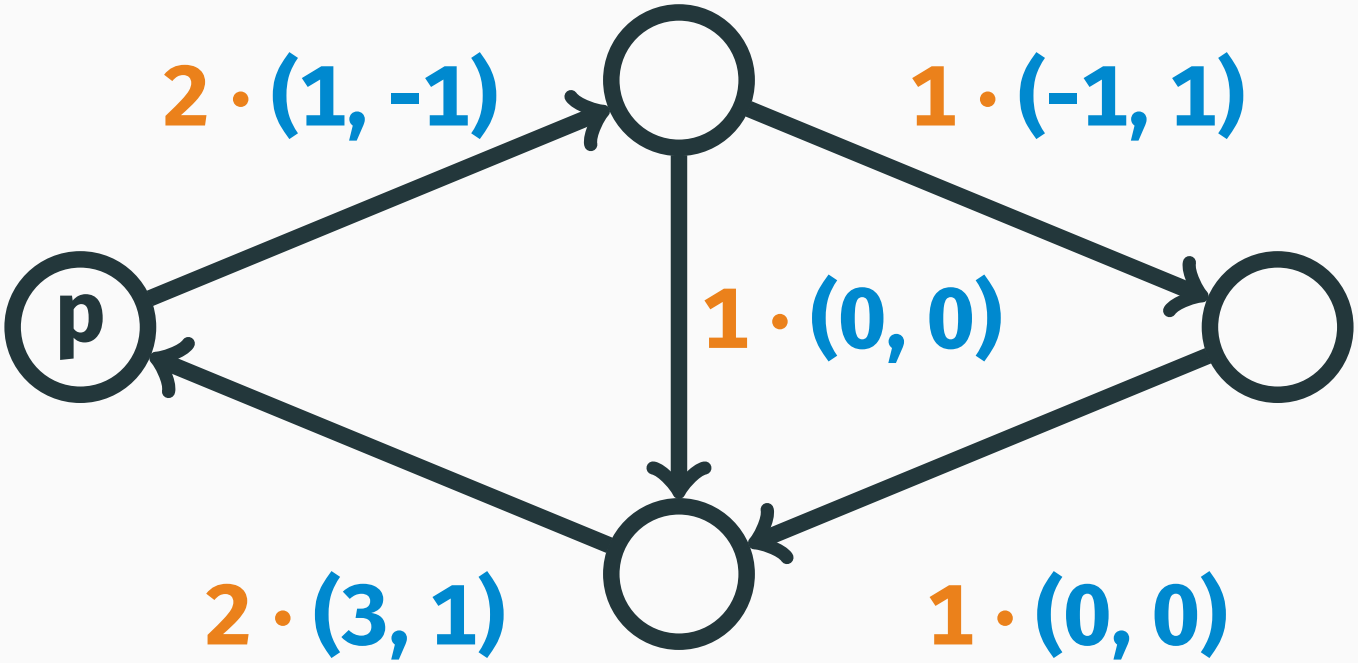


# G[$x_t$ > 0] fortement connexe



# Relaxations

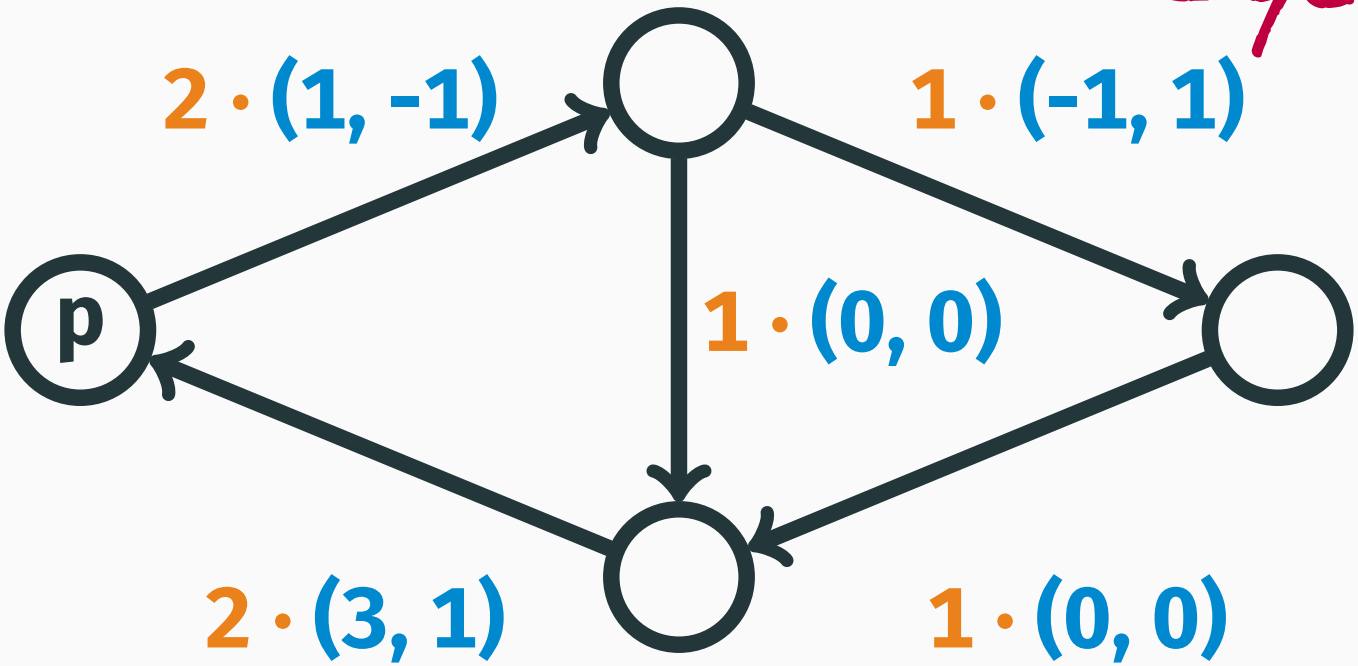
$$\sum_{t \in \text{entrantes}(q)} x_t = \sum_{t \in \text{sortantes}(q)} x_t$$



# Relaxations

$$\sum_{t \in \text{entrantes}(q)} x_t = \sum_{t \in \text{sortantes}(q)} x_t$$

Cycle eulérien



# Relaxations

- Définissable avec une courte formule de l'arithmétique de Presburger
- Rapide en pratique et permet de prouver l'absence de bogues
- Approche similaire avec $\rightsquigarrow_{\mathbb{R}_{\geq 0}}$ et l'arithmétique linéaire $\mathbf{FO}(\mathbb{R}, +)$

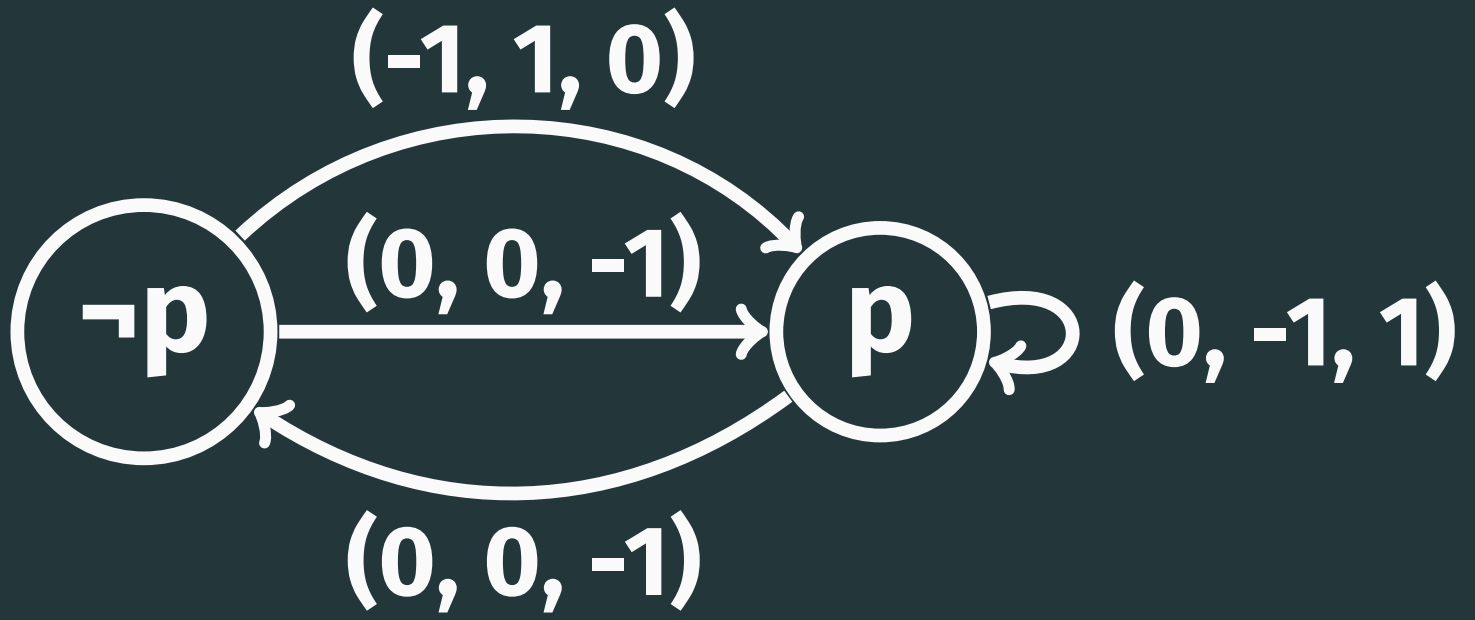
# Relaxations

- Définissable avec une courte formule de l'arithmétique de Presburger
- Rapide en pratique et permet de prouver l'absence de bogues
- Approche similaire avec $\rightsquigarrow_{\mathbb{R}_{\geq 0}}$ et l'arithmétique linéaire $\mathbf{FO}(\mathbb{R}, +)$

# Relaxations

- Définissable avec une courte formule de l'arithmétique de Presburger
- Rapide en pratique et permet de prouver l'absence de bogues
- Approche similaire avec $\rightsquigarrow_{\mathbb{R}_{\geq 0}}$ et l'arithmétique linéaire $\mathbf{FO}(\mathbb{R}, +)$

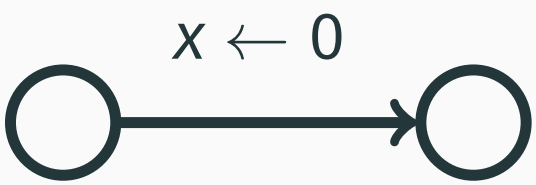
# Exemple



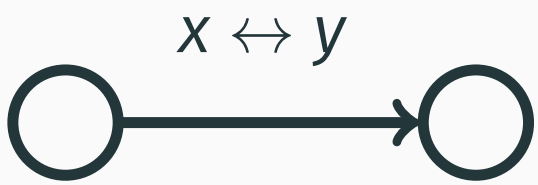

$\neg p(5, 0, 0) \rightsquigarrow p(0, 0, 0)$?

# Autres opérations communes en modélisation:

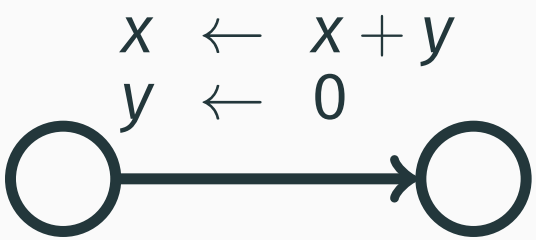
Remise à zéro



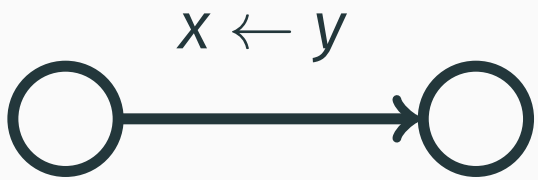

Échange




Transfert




Copie



# Autres opérations communes en modélisation:

Remise à zéro

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Échange

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Transfert

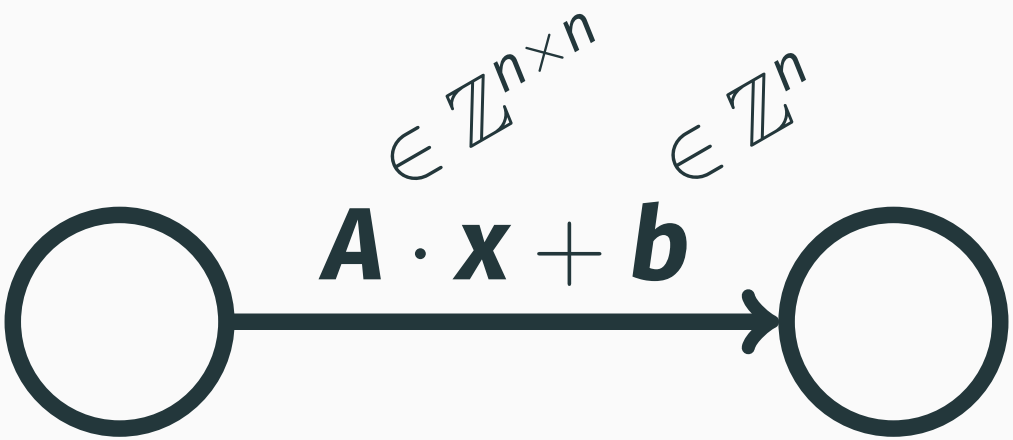
$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Copie

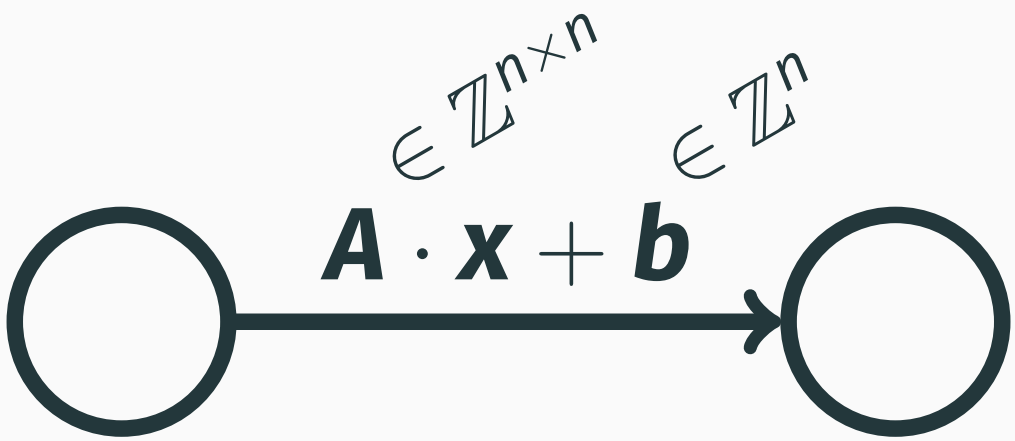
$$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Transformations affines

# Systèmes avec opérations affines:

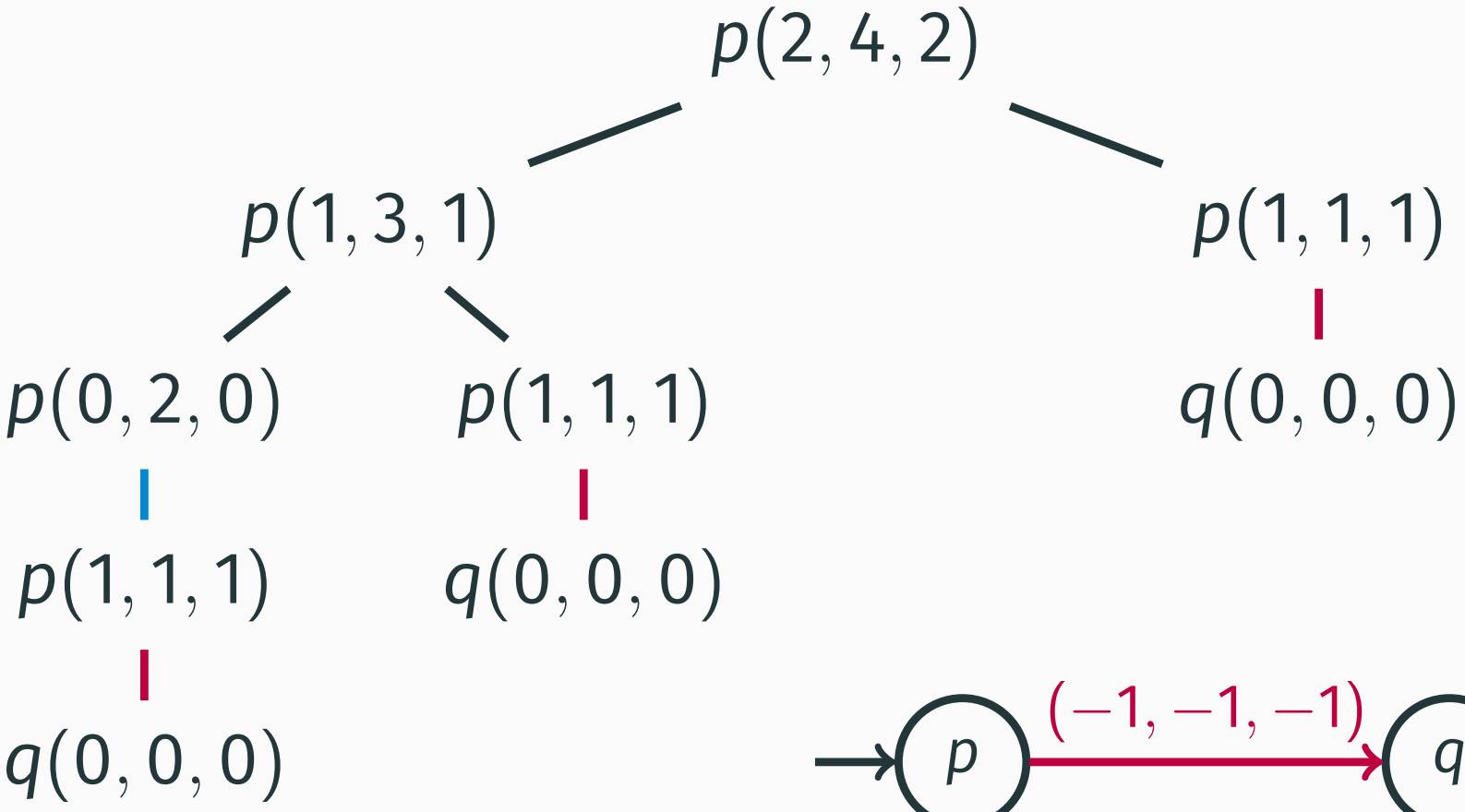


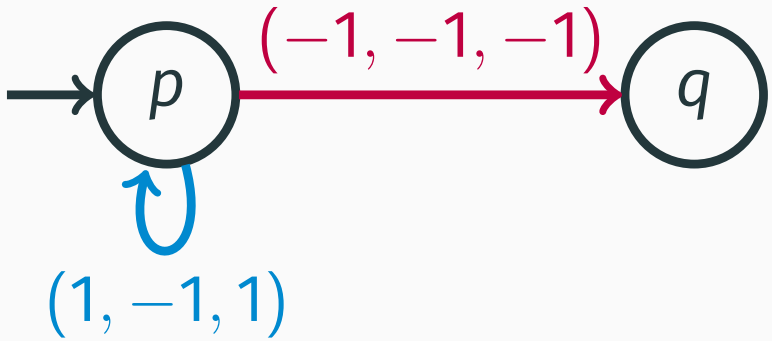
# Systèmes avec opérations affines:



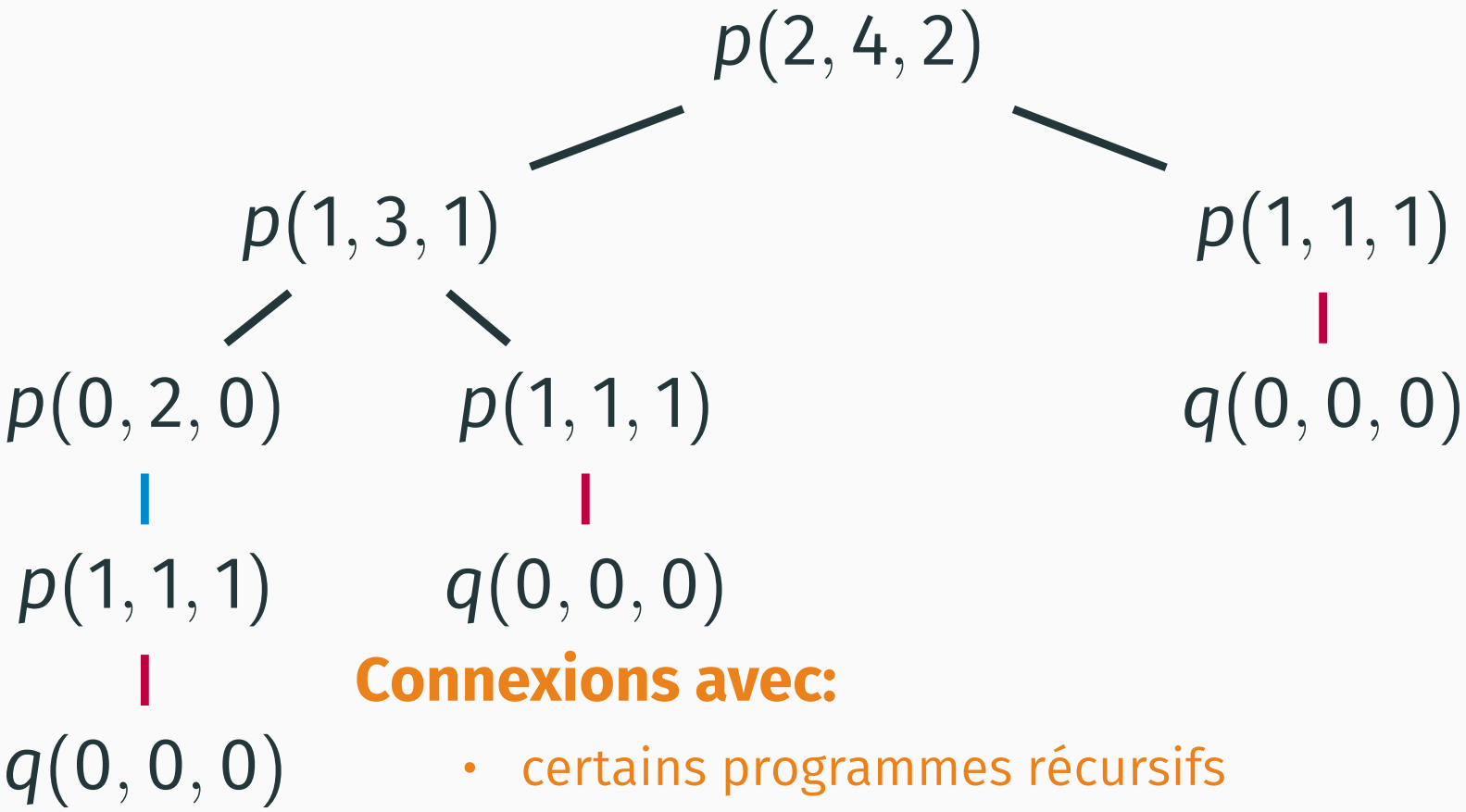

$\rightsquigarrow_{\mathbb{Z}}$ définissable en arithmétique de Presburger si monoïde $\langle A_t : t \in T \rangle$ est fini

## Récursion modélisable avec branchement:



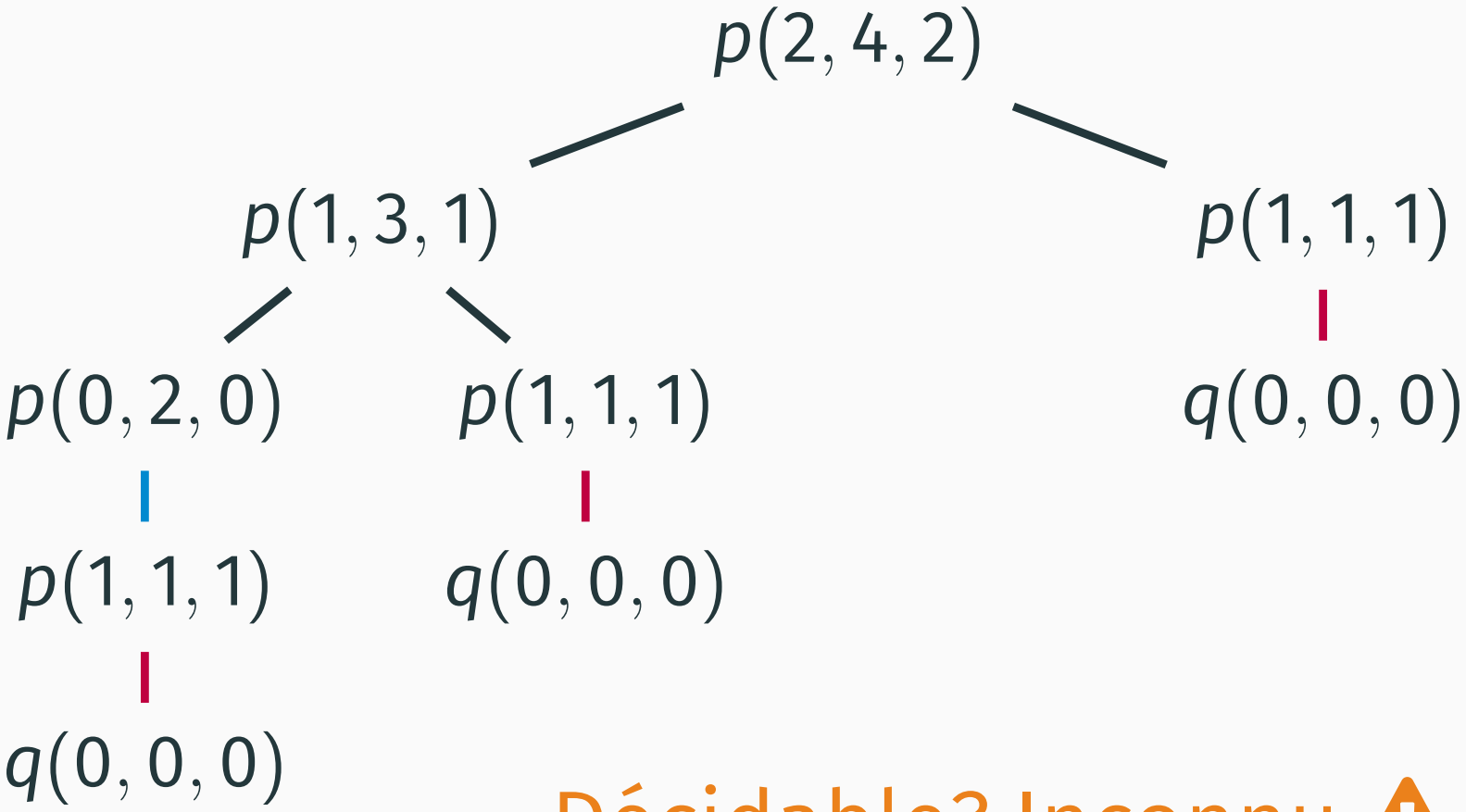
## Récursion modélisable avec branchement:




**Connexions avec:**

- certains programmes récursifs
- modèles de linguistique computationnelle
- analyse de protocoles cryptographiques
- langages de requête de bases de données

## Récursion modélisable avec branchement:




Décidable? Inconnu ⚠

# Conclusion

- **Machines à compteurs** $\equiv$ programmes
- **Systèmes d'addition de vecteurs**
  modélisent certains programmes
- **Complexité colossale** contournable en
  pratique avec **solveurs logiques**
- **Pub:** IGL752 – Techniques de vérification
  IFT503 – Théorie du calcul

# Merci!