*Maths en mouvement*


# Topologie du consensus

Christine Tasson
tasson@pps.jussieu.fr

*le 4 mai 2011*

# Les systèmes distribués en questions

Qu'est ce que c'est ? Un ensemble de processeurs (calculateurs, postes informatiques,. . .) indépendants, connectés par un réseau de communication.

Qu'est ce que c'est ? Un ensemble de processeurs (calculateurs, postes informatiques,. . .) indépendants, connectés par un réseau de communication.

## À quoi ça sert ?

- Accélérer les calculs.

Qu'est ce que c'est ? Un ensemble de processeurs (calculateurs, postes informatiques,. . .) indépendants, connectés par un réseau de communication.

## À quoi ça sert ?

- Accélérer les calculs.
- Être tolérant aux pannes.

Qu'est ce que c'est ? Un ensemble de processeurs (calculateurs, postes informatiques,. . .) indépendants, connectés par un réseau de communication.

## À quoi ça sert ?

- Accélérer les calculs.
- Être tolérant aux pannes.
- Travailler à distance sur une même opération.




# Le problème du consensus

Problématique : $n$ processus indépendants communiquent en s'envoyant des messages, ont la possibilité de faire des calculs et possèdent un registre de valeur.

Trouver un algorithme implémenté pour chaque processus, tel que le registre

- au départ, contient une valeur initiale
- une fonction de transition fait évoluer le registre en fonction des messages reçus
- à la fin, le registre contient une valeur de décision

Les valeurs de décisions de chaque processus doivent être les mêmes, quelles que soient les valeurs initiales, quel que soit l'ordonnancement.

Questions :

- Terminaison ?
- Processus indépendants ? (pas de synchronisation,...)

Questions :

- Terminaison ?
- Processus indépendants ? (pas de synchronisation,...)

**Algorithme**

```
d = "pas_decide"
s = valeur_initiale
result = []
while d == "pas_decide" :
    m = receive_message()
    result = transition(result, m)
    d = decide(result)
return le_plus_frequent(result)
```

où les messages contiennent les valeurs initiales et les noms des processus, la fonction `transition` concatène les messages, la fonction `decide` renvoie "decide" lorsque tous les processus apparaissent dans `result`.

Questions :

- Terminaison ?
- Processus indépendants ? (pas de synchronisation,...)

**Algorithme**

```
d = "pas_decide"
s = valeur_initiale
result = []
while d == "pas_decide" :
    m = receive_message()
    result = transition(result, m)
    d = decide(result)
return le_plus_frequent(result)
```

où les messages contiennent les valeurs initiales et les noms des processus, la fonction `transition` concatène les messages, la fonction `decide` renvoie "decide" lorsque tous les processus apparaissent dans `result`. Mais, en présence de pannes...

# Impossibilité du consensus en présence d'une panne

## Problème

Trouver un algorithme tel que : quelles que soient les valeurs initiales, quel que soit l'ordonnancement, quel que soit le processus tombant en panne,

- au moins un processus décide une valeur
- toutes les valeurs de décision sont identiques $D$
- $D$ appartient aux valeurs initiales des processus ayant décidé

## Théorème (Lynch, Fischer, Patterson 85)

*Il n'existe pas de protocole correct pour le problème du consensus en présence d'au plus un crash.*

# Impossibilité du consensus en présence d'une panne

## Problème

Trouver un algorithme tel que : quelles que soient les valeurs initiales, quel que soit l'ordonnancement, quel que soit le processus tombant en panne,

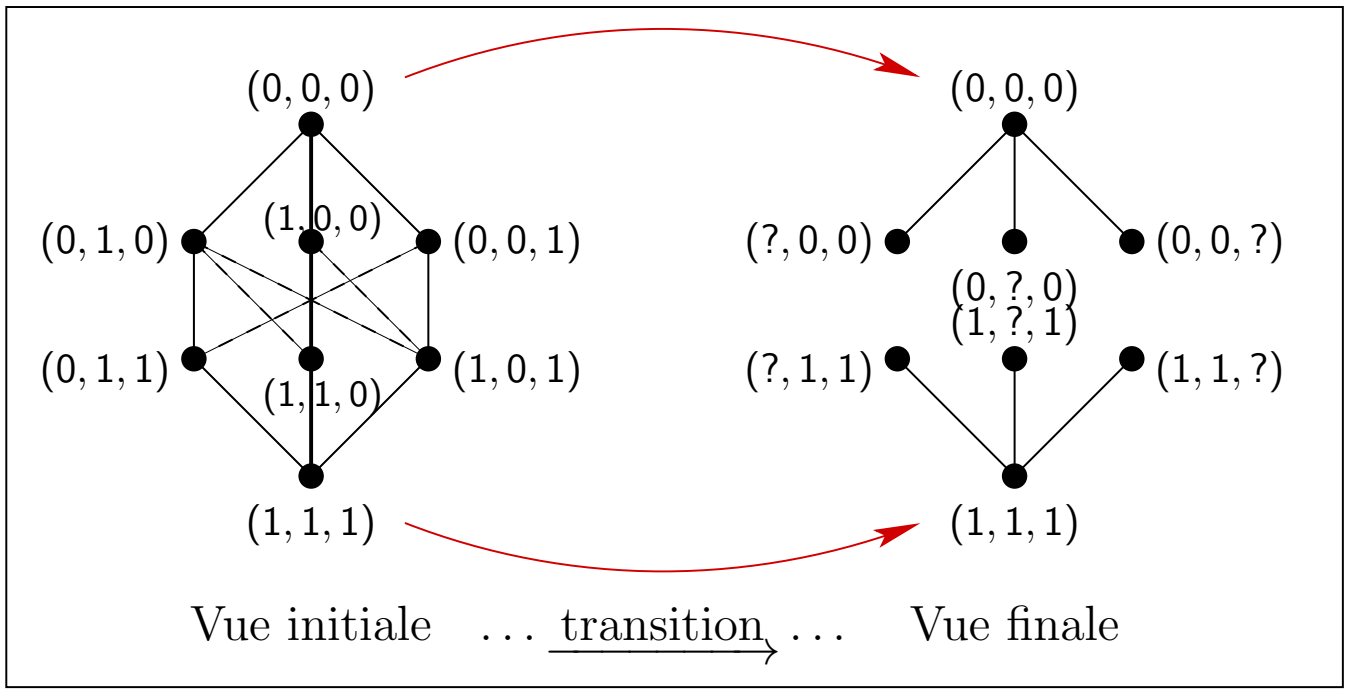
- au moins un processus décide une valeur
- toutes les valeurs de décision sont identiques $D$
- $D$ appartient aux valeurs initiales des processus ayant décidé

## Théorème (Lynch, Fischer, Patterson 85)

*Il n'existe pas de protocole correct pour le problème du consensus en présence d'au plus un crash.*

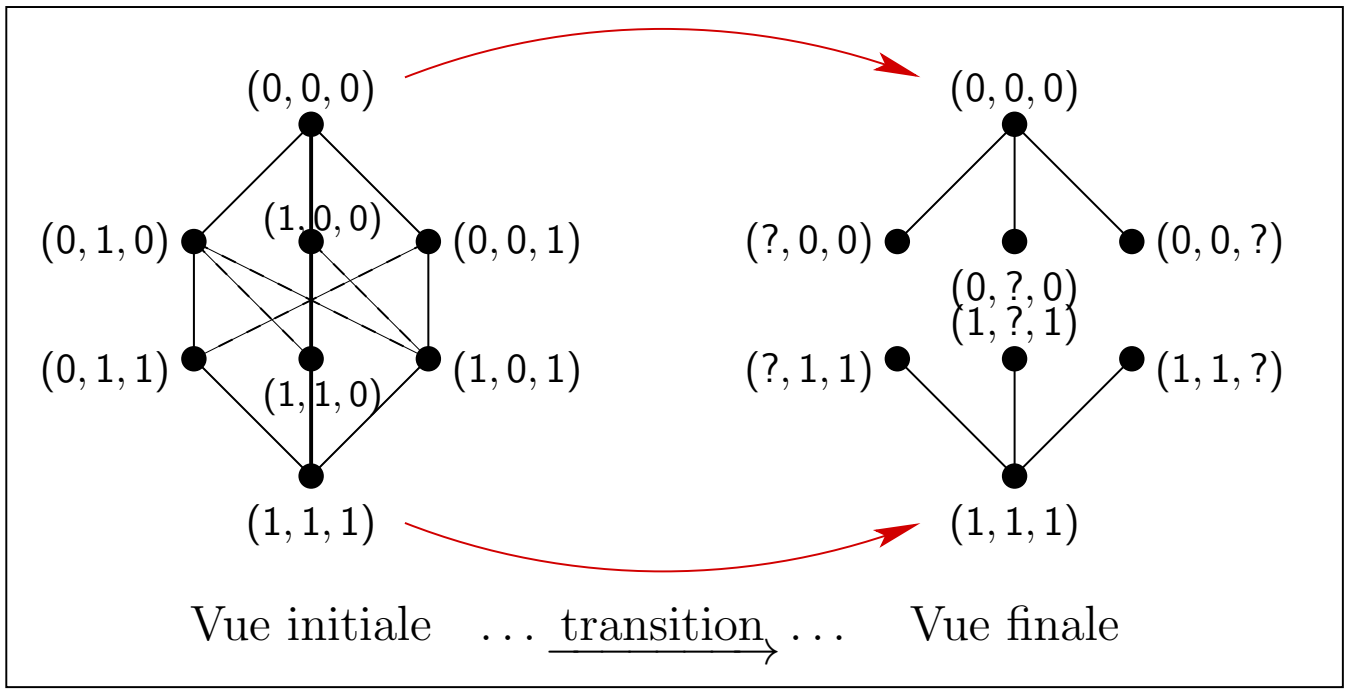
- Ne signifie pas que le problème n'a pas de solution pratique
- Mais, besoin d'hypothèses plus fortes sur le modèle

# Principe de la preuve d'impossibilité



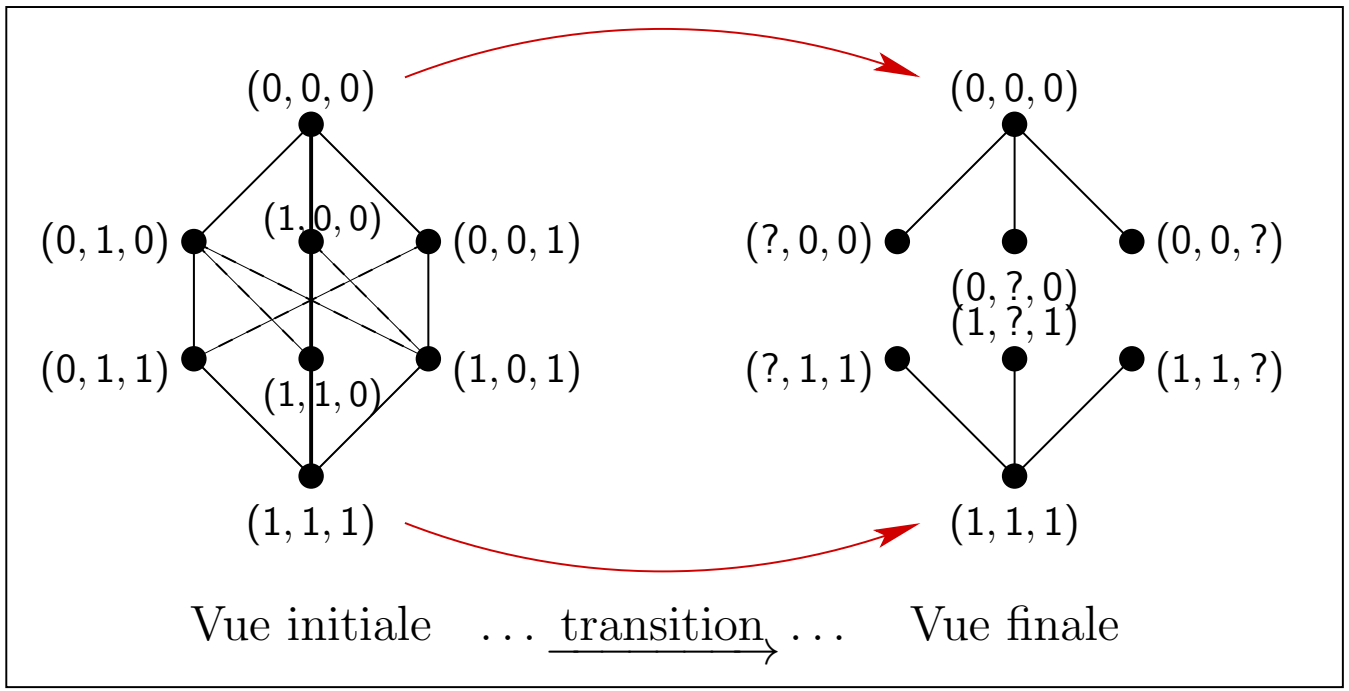

- La valeur de décision appartient aux valeurs initiales
- L'algorithme doit fonctionner quelle que soit la panne

# Principe de la preuve d'impossibilité



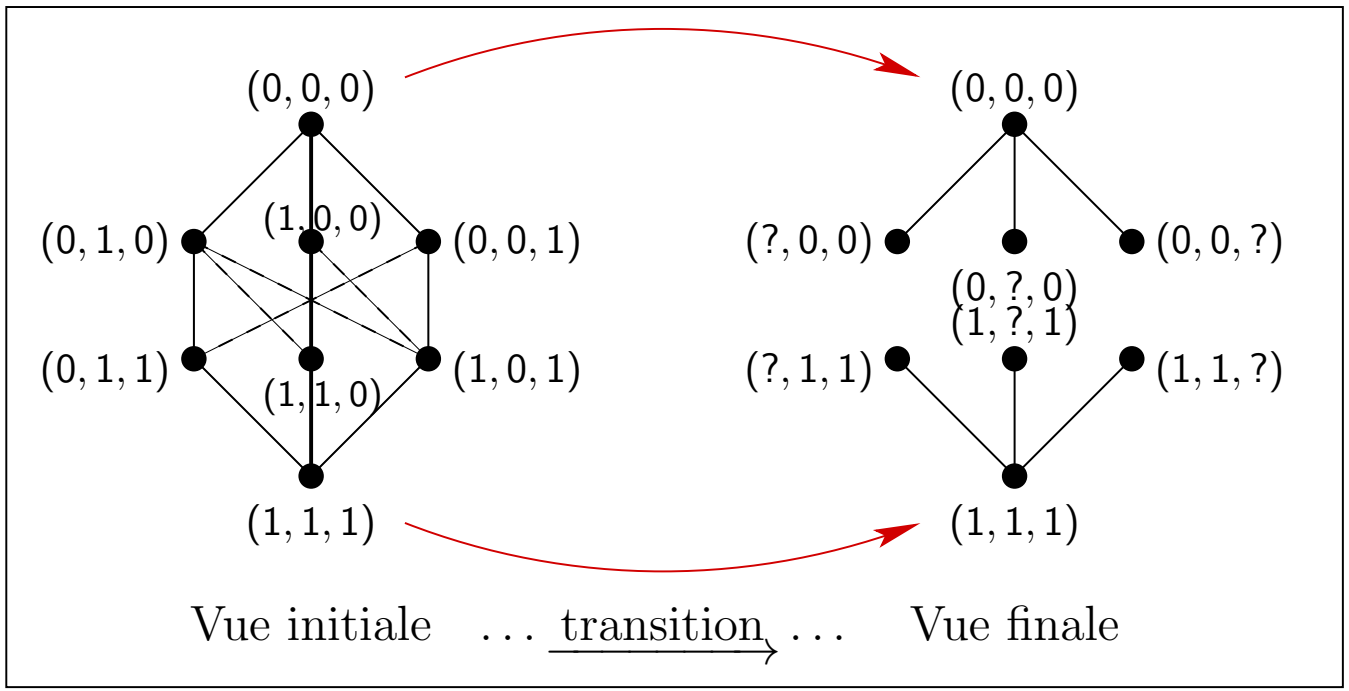

- La valeur de décision appartient aux valeurs initiales
  $\Rightarrow (0,0,0)$ et $(1,1,1)$ sont respectivement envoyés sur les blocs supérieurs et inférieurs.
- L'algorithme doit fonctionner quelle que soit la panne

# Principe de la preuve d'impossibilité




- La valeur de décision appartient aux valeurs initiales
  $\Rightarrow (0,0,0)$ et $(1,1,1)$ sont respectivement envoyés sur les blocs supérieurs et inférieurs.
- L'algorithme doit fonctionner quelle que soit la panne
  $\Rightarrow (0,0,0)$ et $(1,1,1)$ sont envoyés sur la même valeur finale,. . .

# Principe de la preuve d'impossibilité



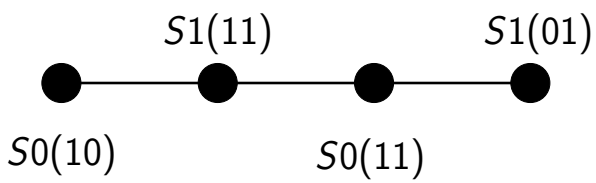

- La valeur de décision appartient aux valeurs initiales
  $\Rightarrow (0,0,0)$ et $(1,1,1)$ sont respectivement envoyés sur les blocs supérieurs et inférieurs.
- L'algorithme doit fonctionner quelle que soit la panne
  $\Rightarrow (0,0,0)$ et $(1,1,1)$ sont envoyés sur la même valeur finale,...
  $\Rightarrow$ CONTRADICTION

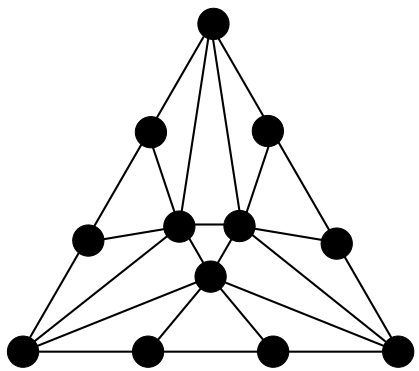
# Topologies combinatoire et algébrique dirigée

Problématique : $n$ processus partagent une mémoire partitionnée (une case dédiée à chaque processus). Chaque processus écrit dans sa case, puis lit le contenu de la totalité de la mémoire.

Solution discrète : [Herlihy, Rajsbaum, Shavit, Saks, Zaharoglou]

- Raisonner sur la géométrie et la topologie d'objets combinatoires :



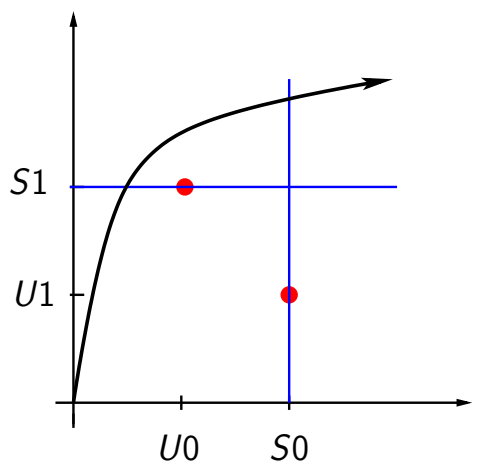



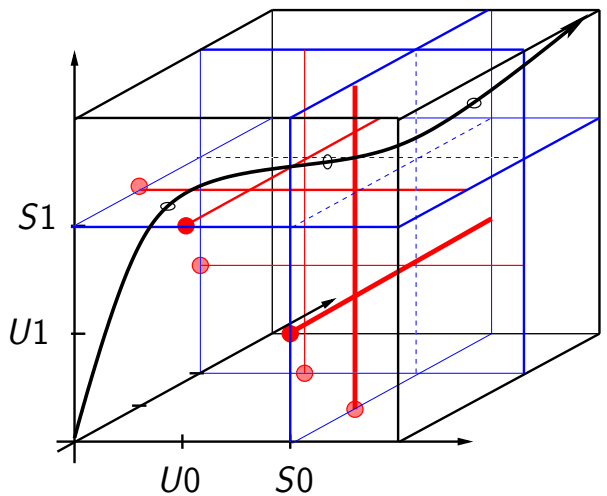
- Le consensus est impossible.

Problématique : $n$ processus partagent une mémoire partitionnée (une case dédiée à chaque processus). Chaque processus écrit dans sa case, puis lit le contenu de la totalité de la mémoire.

L'approche continue : [Grandis, Raussen, Goubault]

- Raisonner sur la géométrie et la topologie d'objets continus :

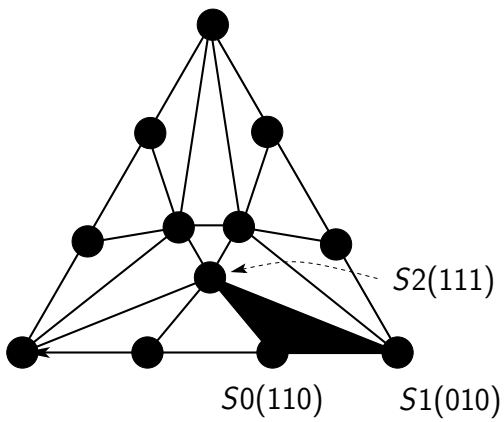


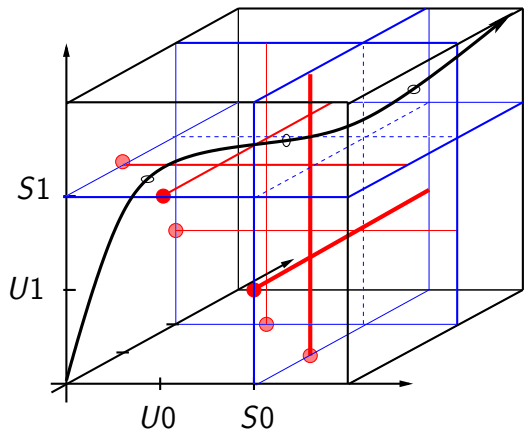




- Topologie algébrique dirigée.

Problématique : $n$ processus partagent une mémoire partitionnée (une case dédiée à chaque processus). Chaque processus écrit dans sa case, puis lit le contenu de la totalité de la mémoire.

Relier les approches discrète et continue :




**Théorème**

*Les simplexes de l'objet combinatoire correspondent aux classes d'homotopie des chemins dirigés.*