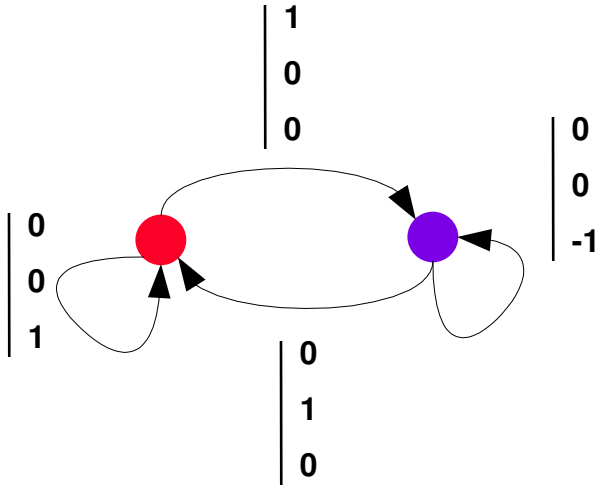
Équations récurrentes uniformes

Pour $1 \leq i,j,k \leq n$

$a(i,j,k) = b(i,j-1,k) + a(i,j,k-1)$

$b(i,j,k) = a(i-1,j,k) + b(i,j,k+1)$



- Description à assignation unique.
- Dépendances uniformes.
- Principe de calcul: membre droit d'abord.

- Dépendances explicites.
- Ordre d'exécution implicite.
- Mémoire dépliée.

SUREs: principes généraux

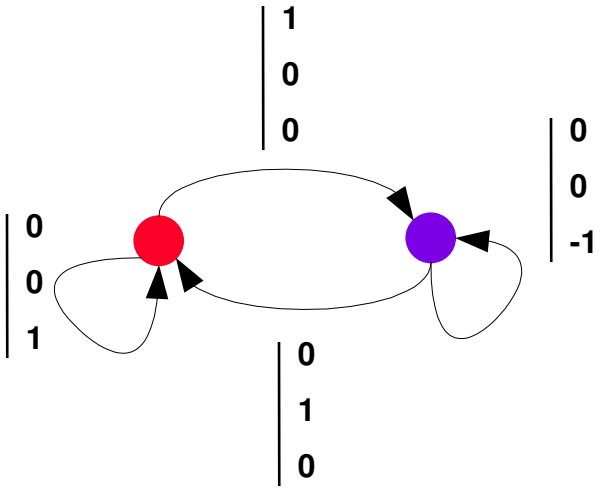
- Analyse des unions de cycles de poids total nul:
  - → calculabilité du système.
  - → degré de parallélisme du système.
- Analyse “duale” (en termes de prog. linéaire):
  - → ordonnancement du système.
- Attribution d’une sémantique temps + espace:
  - → description d’une architecture systolique lorsque le “temps” est mono-dimensionnel.
  - → pas de mémoire globale mais des temporisations.

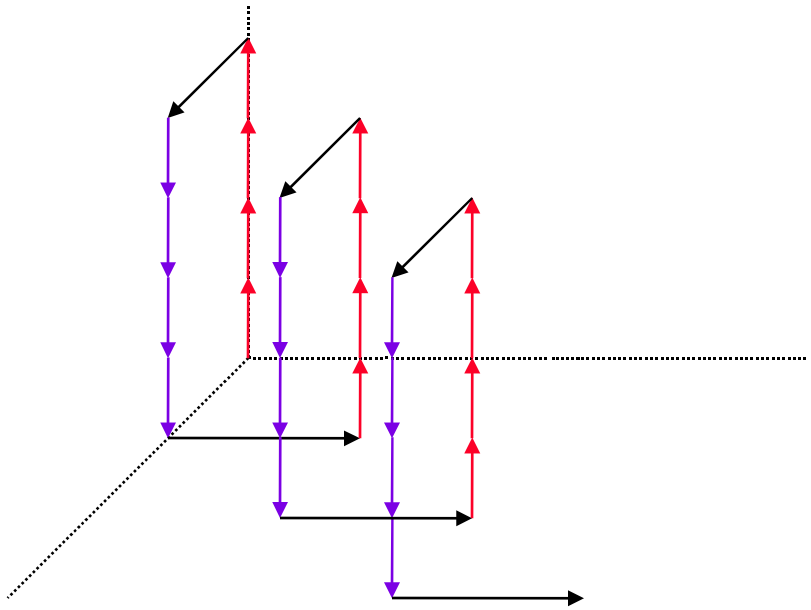
SURE, exemple

Pour $1 \leq i,j,k \leq n$

$a(i,j,k) = b(i,j-1,k) + a(i,j,k-1)$

$b(i,j,k) = a(i-1,j,k) + b(i,j,k+1)$





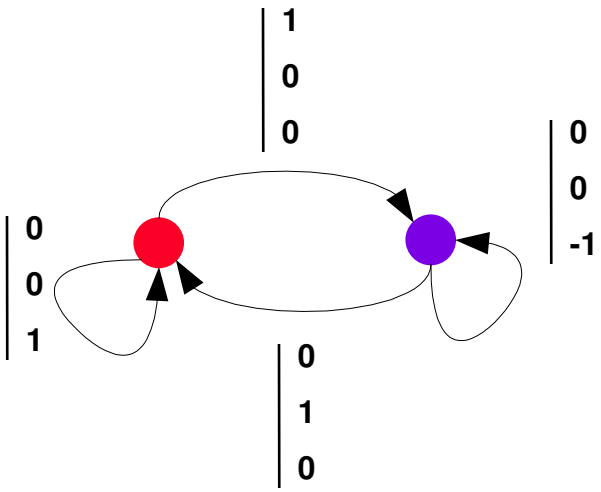
chemin de dépendance en N*N

SURE, exemple

Pour $1 \leq i,j,k \leq n$

$a(i,j,k) = b(i,j-1,k) + a(i,j,k-1)$

$b(i,j,k) = a(i-1,j,k) + b(i,j,k+1)$




```
do i = 1,n
  do k = n,1,-1
    dopar j = 1,n
      b(i,j,k) = a(i-1,j,k) + b(i,j,k+1)
    enddo
  enddo
  do k = 1,n
    dopar j = 1,n
      a(i,j,k) = b(i,j-1,k) + a(i,j,k-1)
    enddo
  enddo
enddo
```

Pourquoi ce modèle?

- Avantages:
  - Modèle simplifié, plus simple à analyser.
  - Flot de calcul explicite. Correspondance calcul-mémoire.
  - Dépendances uniformes → “délais” constants.
  - Description “propre” à la fois proche de l’algorithme et de l’architecture.
  - Possibilités de transformations dans le même formalisme.
- Inconvénients:
  - Langage de description correspondant restrictif.
  - Langage loin des habitudes des programmeurs.