Algorithmes distribuées auto-stabilisants.

Johanne Cohen[1]

[1]LORIA/CNRS, Nancy, France.

Références

Shlomi Dolev
Self-Stabilization

E.W. Dijkstra
Self-stabilizing systems in spite of distributed control.
Commun. ACM 17 (1974), 11 : 643-644.

Gérard Tel
Introduction to Distributed Algorithms

Sébastien Tixieul
Introduction to Self-stabilization
http ://sogea.loria.fr/.

Plan

Petit exemple (1/2).

Considérons la suite u_n suivante

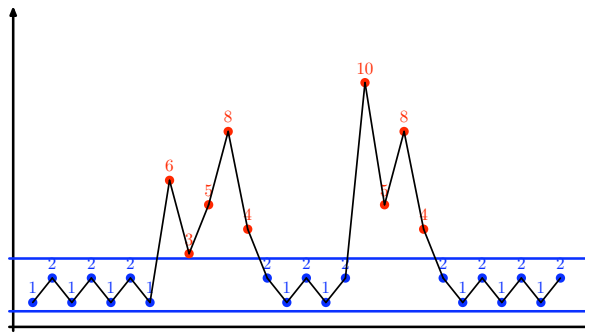
- $u_0 = x$
- $u_n = \begin{cases} \frac{u_n}{2} & si \quad u_n \ pair \\ \frac{3u_n+1}{2} & sinon \quad (u_n \ impair) \end{cases}$

n	0	1	2	3	4	5	6	7	8	9	10	11
u_n	17	26	13	20	10	5	8	4	2	1	2	1

Quelque soit la valeur initiale, elle converge vers le comportement :

1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2

Petit exemple (2/2).



Plan

Plan

Rappel de définition

Definition
Un *système de transition* est un couple $S = (C, \rightarrow)$ où

C est un ensemble de configurations

$\rightarrow$ est une relation binaire sur C

Definition
Une *exécution* de S est une suite maximale (finie ou infinie) $E = (\gamma_0, \gamma_1, \ldots, \gamma_n, \ldots)$ telle que

pour tout $i \geq 0$, on a $\gamma_i \rightarrow \gamma_{i+1}$

Definition
Une *spécification* est un prédicat P sur l'ensemble des exécutions

Retour au petit exemple

- Ensemble de configurations possibles (N^+)
- fonction de transition $\rightarrow$:

$$\gamma_i \rightarrow \gamma_{i+1} \text{ SSi } \gamma_{i+1} = \begin{cases} \frac{\gamma_i}{2} & si \quad \gamma_i \text{ pair} \\ \frac{3\gamma_i+1}{2} & sinon \quad (\gamma_i \text{ impair}) \end{cases}$$

- une exécution : $E = (17, 26, 13, 20, 10, 5, 8, 4, 2, 1, 2, 1)$
- Un prédicat $P(\gamma) = (\gamma == 1) \vee (\gamma == 2)$

Définition de l'auto-stabilisation

un système est auto-stabilisant est défini par Dijkstra en 1973

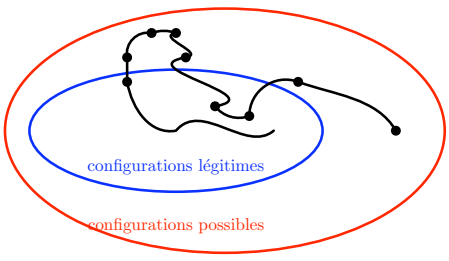
> `Regardless of its initial state, it is guaranteed to arrive at a legitimate state in a finite number of steps.`

Plus formellement,

Definition
Un système de transition $S = (C, \rightarrow)$ est *auto-stabilisant* pour une spécification P s'il existe un ensemble $L \subseteq C$ de configurations légitimes vérifiant les propriétés suivantes

1. `Convergence` : toute exécution atteint une configuration de L.
2. `Correction` : toute exécution débutant par une configuration de L satisfait P

Définitions sur les configurations



Complexité d'un algorithme auto-stabilisant

en espace : est la taille (en nombre de bits) des variables utilisées localement pour chaque processeur.

en temps : est le nombre de tours nécessaires pour parvenir à une configuration légitime (au cours d'un tour, chaque processeur exécute ou tente d'exécuter son code local).

Retour au petit exemple

- Ensemble de configurations possibles (N^+)
- fonction de transition $\rightarrow$:

$$\gamma_i \rightarrow \gamma_{i+1} \text{ SSi } \gamma_{i+1} = \begin{cases} \frac{\gamma_i}{2} & \textit{si} \quad \gamma_i \textit{ pair} \\ \frac{3\gamma_i+1}{2} & \textit{sinon} \quad (\gamma_i \textit{ impair}) \end{cases}$$

- Un prédicat $P(\gamma) = (\gamma == 1) \vee (\gamma == 2)$
- Configurations légitimes : $u_n = 1$, $u_n = 2$, i.e : $L = \{1, 2\}$
- propriété de `correction`
- propriété de `convergence` ? (i.e : toute exécution atteint une configuration de L.)

Conjecture à 1 000 000 Dollars pour la prouver.

Retour au petit exemple

- Ensemble de configurations possibles (N^+)
- fonction de transition $\rightarrow$:

$$\gamma_i \rightarrow \gamma_{i+1} \text{ SSi } \gamma_{i+1} = \begin{cases} \frac{\gamma_i}{2} & \textit{si} \quad \gamma_i \textit{ pair} \\ \frac{3\gamma_i+1}{2} & \textit{sinon} \quad (\gamma_i \textit{ impair}) \end{cases}$$

- Un prédicat $P(\gamma) = (\gamma == 1) \vee (\gamma == 2)$
- Configurations légitimes : $u_n = 1$, $u_n = 2$, i.e : $L = \{1, 2\}$
- propriété de `correction` ? (i.e toute exécution débutant par une configuration de L satisfait P)

Si $\gamma_i = 1$ alors $\gamma_{i+1} = 2$
Si $\gamma_i = 2$ alors $\gamma_{i+1} = 1$

- propriété de `convergence` ? (i.e : toute exécution atteint une configuration de L.)

Conjecture à 1 000 000 Dollars pour la prouver.

Retour au petit exemple

- Ensemble de configurations possibles (N^+)
- fonction de transition $\rightarrow$:

$$\gamma_i \rightarrow \gamma_{i+1} \text{ SSi } \gamma_{i+1} = \begin{cases} \frac{\gamma_i}{2} & \textit{si} \quad \gamma_i \textit{ pair} \\ \frac{3\gamma_i+1}{2} & \textit{sinon} \quad (\gamma_i \textit{ impair}) \end{cases}$$

- Un prédicat $P(\gamma) = (\gamma == 1) \vee (\gamma == 2)$
- Configurations légitimes : $u_n = 1$, $u_n = 2$, i.e : $L = \{1, 2\}$
- propriété de `correction` : OUI
- propriété de `convergence` ? (i.e : toute exécution atteint une configuration de L.)

Conjecture à 1 000 000 Dollars pour la prouver.

Retour au petit exemple

- Ensemble de configurations possibles (N^+)
- fonction de transition $\rightarrow$:

$$\gamma_i \rightarrow \gamma_{i+1} \text{ SSi } \gamma_{i+1} = \begin{cases} \frac{\gamma_i}{2} & \textit{si} \quad \gamma_i \textit{ pair} \\ \frac{3\gamma_i+1}{2} & \textit{sinon} \quad (\gamma_i \textit{ impair}) \end{cases}$$

- Un prédicat $P(\gamma) = (\gamma == 1) \vee (\gamma == 2)$
- Configurations légitimes : $u_n = 1$, $u_n = 2$, i.e : $L = \{1, 2\}$
- propriété de `correction` : OUI
- propriété de `convergence` ? (i.e : toute exécution atteint une configuration de L.)

Conjecture à 1 000 000 Dollars pour la prouver.

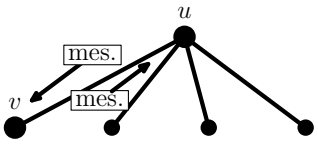
Notation

Commande

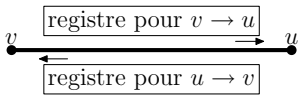
`règle gardée` $\rightarrow$ `instruction`

- une `règle gardée` : un prédicat sur les états des voisins.
- une action `instruction` : un prédicat sur les états des voisins.
- `instruction` : est exécutée si la `règle gardée` est vraie.

Communications par envoi de messages.



Communications par registres partagés.

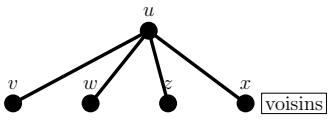


Communications par mémoire partagée.

Definition

A chaque pas atomique, un processus

- peut lire les états de tous ses voisins
- peut modifier son état.



Plan

l'algorithme auto-stabilisant du jeton

- Contexte :
  un anneau de taille N où chaque noeud a un voisin à GAUCHE et à DROITE.
- Objectif : exclusion mutuelle.
  un unique jeton qui circule indéfiniment et de manière équitable

Hypothèse

- Communications par mémoire partagée.
- Tous les processeurs sont activés par un ordonnanceur appelé DÉMON
- Ici, le démon est CENTRALISÉ
  - Il active un processeur à la fois pour faire une séquence atomique d'instructions.
- Le démon centralisé est ÉQUITABLE
  - Il active un processeur une infinité de fois pour toutes exécutions infinies.

Idée de l'algorithme

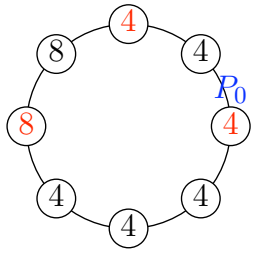
- Chaque site i possède une valeur : σ_i
- La présence du jeton dépendra de la différence entre sa valeur et son prédécesseur (voisin de GAUCHE)
- Déplacement du jeton = la "propagation" de vagues

Algorithme Dijkstra

- Soit $k \in N^+$ avec $k > n$ avec n le nombre de processeurs
- Initialisation sur chaque machine i :
  - σ_i pris au hasard entre $[0, \ldots, k-1]$
- Définition du jeton
  - P_0 détient le jeton $\Longleftrightarrow \sigma_0 = \sigma_{n-1}$
  - P_i détient le jeton $\Longleftrightarrow \sigma_i \neq \sigma_{i-1}$

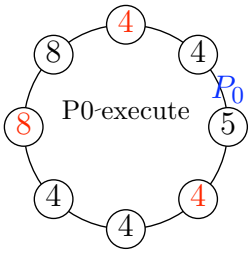
- `Exécution du code :`
  - `pour` P_0 `:` $\sigma_0 = \sigma_{n-1} \rightarrow \sigma_0 := \sigma_0 + 1 \mod k$
  - `pour` P_i `:` $\sigma_i \neq \sigma_{i-1} \rightarrow \sigma_i := \sigma_{i-1}$

Exemple



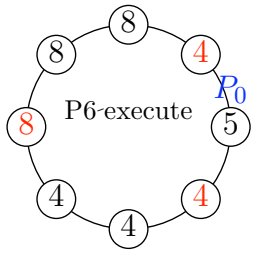
État initial : plusieurs jetons (site en rouge).
Activation de P_0 : Si $\sigma_0 = \sigma_{n-1}$ alors $\sigma_0 := \sigma_0 + 1 \mod k$.

Exemple



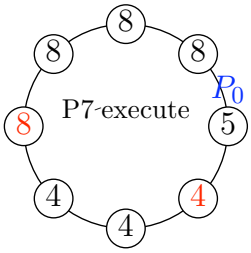
État : plusieurs jetons (site P_1, P_4, P_6 en rouge).
Activation de P_6 : Si $\sigma_6 \neq \sigma_5$ alors $\sigma_6 := \sigma_5$.

Exemple



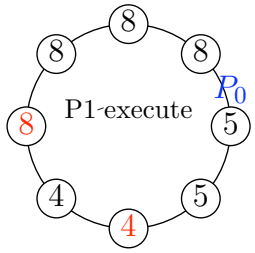
État : plusieurs jetons (site P_1, P_4, P_7 en rouge).
Activation de P_7 : Si $\sigma_7 \neq \sigma_6$ alors $\sigma_7 := \sigma_6$.

Exemple



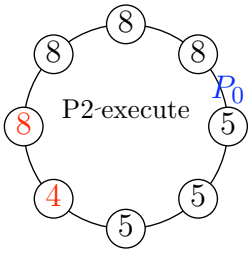
État : plusieurs jetons (site P_1, P_4). Perte d'un jeton
Activation de P_1 : Si $\sigma_1 \neq \sigma_0$ alors $\sigma_1 := \sigma_0$.

Exemple



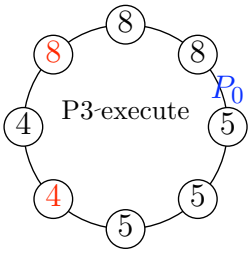
État : plusieurs jetons (site P_2, P_4).
Activation de P_2 : Si $\sigma_2 \neq \sigma_1$ alors $\sigma_2 := \sigma_1$.

Exemple



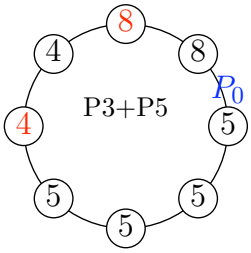
État : plusieurs jetons (site P_3, P_4).
aucune supression de jeton $\forall$ la suite : Activation de P_4.

Exemple



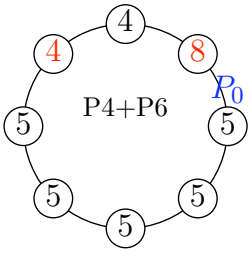
État : plusieurs jetons (site P_3, P_5).
Activation de P_5 et de P_3

Exemple



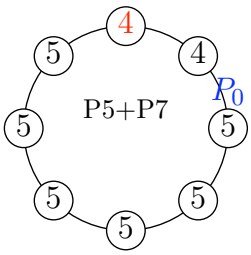
État : plusieurs jetons (site P_4, P_6).
Activation de P_6 et de P_4

Exemple



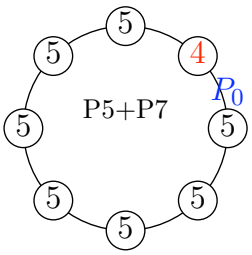
État : plusieurs jetons (site P_5, P_7).
Activation de P_7 et de P_5

Exemple



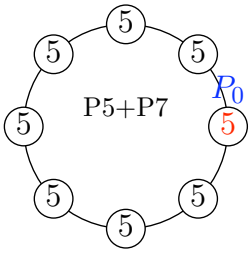
État : 1 jeton sur site P_6.Ici, perte d'un jeton
Configuration légitime Activation de P_6

Exemple



État : 1 jeton sur site P_7.
Configuration légitime Activation de P_7

Exemple



État : 1 jeton sur site P_0.
Configuration légitime Activation de P_0, et ainsi de suite

Remarques.

- Une configuration est un vecteur de n valeurs entières

$$[\sigma_1, \sigma_2, \ldots, \sigma_n]$$

- Séquence atomique d'instructions = lire $\sigma_{i-1}\oplus$ calculer σ_i
- A un pas donné, un seul processeur est activé à la fois :
  Un seul processeur peut changer son état à la fois.
- Une exécution équitable est une suite de configurations
  1. dans laquelles exactement un processeur change d'état.
  2. dans laquelles chaque processeur change d'état une infinité de fois pour toute exécution infinie.

Preuve (1/4)

Proposition 1
Il n'existe pas de configuration sans jeton.

Démonstration.
Si aucun des P_i n'a de jeton, alors $\sigma_0 = \sigma_1 = \cdots = \sigma_{n-1}$. D'où la contradiction □

Proposition 2
Il n'y a pas de configuration terminale (plus de règles activées).

Démonstration.
Soit L l'ensemble des configurations ayant au moins 1 jeton.
Toute activation d'une règle fait passer le jeton d'une machine à la suivante. □

Preuve (2/4)

Les configurations légitimes sont de la forme tel que $\exists j, \exists v < k$
$\sigma_i = v \ \forall i < j$ et $\sigma_i = v - 1 \mod k \ \forall j \leq i \leq n$

Proposition 3 :
Soit c une configuration contenant un seul jeton, alors le système converge vers L (l'ensemble des configurations ayant 1 jeton).

Démonstration.
Soit P_i le processeur capable d'activer une règle.
Après son activation, seul le processeur P_{i+1} est capable d'activer la règle et ainsi de suite...

□

Preuve (3/4).

Proposition 4 :
Dans une exécution infinie, P_0 est activé une infinité de fois.

Démonstration.

- $S(t) = \{i : i \neq 0 \wedge P_i \text{ a le jeton}\}$ et $F(t) = \sum_{i \in S(t)}(N - i)$
- A chaque pas n'impliquant pas de P_0, $F(t+1) < F(t)$: car le jeton se déplace de 1 ou il disparaît.
- $0 \leq F(t)$
- Donc dans une exécution infinie, P_0 est activé une infinité de fois.

□

Preuve (4/4).

Proposition 5 :
Le système converge vers L, l'ensemble des configurations ayant exactement 1 jeton.

Démonstration.

- Il reste à prouver que pour une exécution infinie, l'algorithme visite une configuration légitime.
- Soit c la configuration où P_0 est activé pour la 1ere fois.
- Avant c, $\sigma_0 = \sigma_{n-1} = y$, et dans c, $\sigma_0 = y + 1 \mod k$
- Quand P_0 de nouveau activé ? quand $\sigma_{n-1} = y + 1$ via σ_{n-2}
- Comme $n > 2$, il y a deux processeurs ayant le m̂ numéro.
- Au fur et à mesure que P_0 est activé, le nombre de $\neq$ numéros diminue.

□

Plan

Technique de preuve

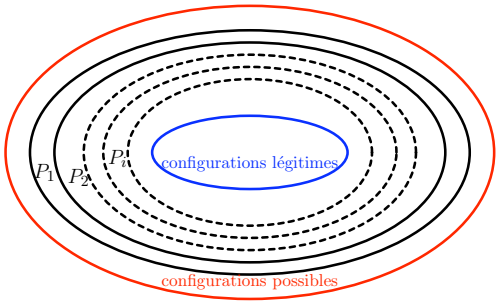
La méthode de la fonction de potentiel :

- Définir une fonction $\mathcal{POIDS}$: ens des configurations $\rightarrow N$ telle que
  - pour toute configuration légitime L, on a $\mathcal{POIDS}(L) = 0$.
  - l'exécution de l'algorithme $\mathcal{A}$ provoque une décroissance.

    Si $c \rightarrow_{\mathcal{A}} c'$ alors $\mathcal{POIDS}(c) > \mathcal{POIDS}(c')$

Méthode par raffinement

Preuve basée que l'exécution est composée de configurations satisfaisant P_1, puis P_2, ..., ensuite P_i,



Plan

Distance dans un graphe

Hypothèse

- $G = (V, E)$ un graphe, et un sommet u appelé RACINE
- $\Gamma(i)$: ensemble des voisins de i dans G
- Communication de message : mémoire partagée.

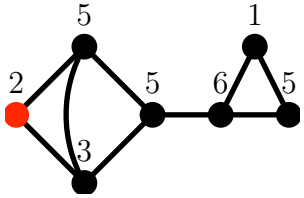
Objectif de v : trouver la distance entre u et v

Algorithme

- pour u : $true \rightarrow Distance_i := 0$
- pour $v \neq u$: $true \rightarrow Distance_i := \min_{j \in \Gamma(i)}\{Distance_j + 1\}$

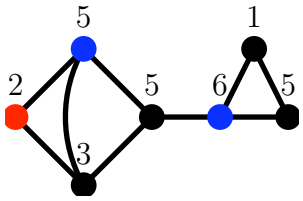
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



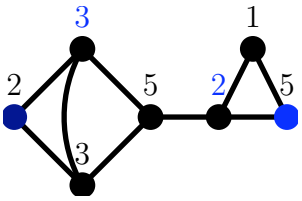
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



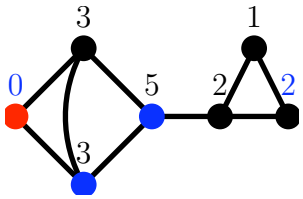
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



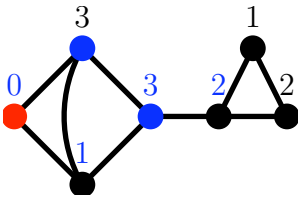
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



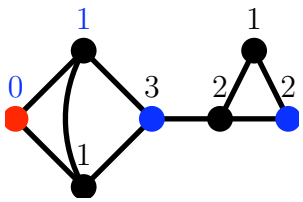
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



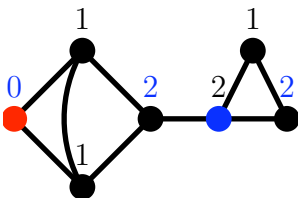
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



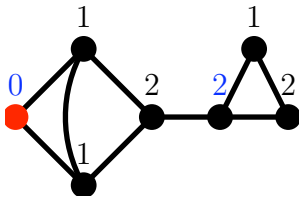
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



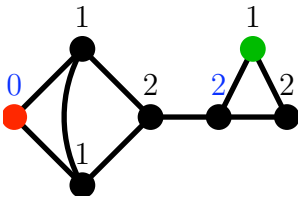
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



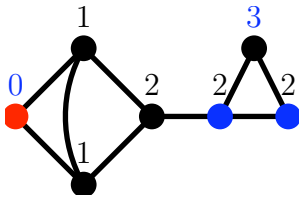
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



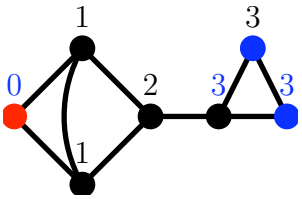
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



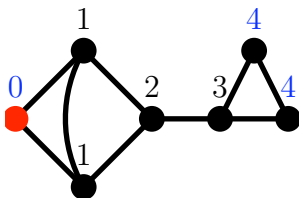
Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



Exemple

Graphe dont les étiquettes correspondent à la distance entre le sommet rouge et tous les autres sommets du graphe.



Preuve par rafinement

Lemme
A partir d'une configuration de départ $\mathcal{C}$, l'algorithme converge vers les états légitimes.

Démonstration.
Prédicat P_i : tous les sommets à distance i de u connaissent leur distance entre lui et u.

- Cas de base : $i = 0$. Comme l'exécution est équitable, u est activé au moins une fois dans cette exécution (dans c_0). La valeur est initialisée à la bonne valeur.
- Cas d'induction :
  - Comme l'exécution est équitable, un sommet v à dist. i est activé après de c_{i-1} au moins 1 fois (dans c'_i).
  - Comme v voisin d'un sommet à dist. $i-1$, la distance est réactualisée à la bonne valeur.
  - et ainsi de suite pour tous les sommets à distance i.

□

Preuve par rafinement

Lemme
A partir d'une configuration de départ $\mathcal{C}$, l'algorithme converge vers les états légitimes.

Démonstration.
Prédicat $\mathtt{P_i}$: tous les sommets à distance i de u connaissent leur distance entre lui et u.

- Cas de base : $i = 0$. Comme l'exécution est équitable, u est activé au moins une fois dans cette exécution (dans c_0). La valeur est initialisée à la bonne valeur.
- Cas d'induction :
  - Comme l'exécution est équitable, un sommet v à dist. i est activé après de c_{i-1} au moins 1 fois (dans c'_i).
  - Comme v voisin d'un sommet à dist. $i-1$, la distance est réactualisée à la bonne valeur.
  - et ainsi de suite pour tous les sommets à distance i.

□

Preuve par rafinement

Lemme
A partir d'une configuration de départ $\mathcal{C}$, l'algorithme converge vers les états légitimes.

Démonstration.
Prédicat $\mathtt{P_i}$: tous les sommets à distance i de u connaissent leur distance entre lui et u.

- Cas de base : $i = 0$. Comme l'exécution est équitable, u est activé au moins une fois dans cette exécution (dans c_0). La valeur est initialisée à la bonne valeur.
- Cas d'induction :
  - Comme l'exécution est équitable, un sommet v à dist. i est activé après de c_{i-1} au moins 1 fois (dans c'_i).
  - Comme v voisin d'un sommet à dist. $i-1$, la distance est réactualisée à la bonne valeur.
  - et ainsi de suite pour tous les sommets à distance i.

□

Plan

Définition de la composition

- Soient deux systèmes $S_1 = (C_1, \rightarrow_1)$ et $S_2 = (C_2, \rightarrow_2)$ tels que toutes variables écrites par S_2 n'apparaissent pas dans S_1.

La *composition S de* S_1 *et de* S_2 notée $S_1 \triangleleft S_2$ est un système tel que

- son ensemble des configurations $S_1 \times S_2$
- sa relation de transition est $(\rightarrow_1, \times ID) \cup (ID \times \rightarrow_2)$

Théorème.

- Soit S_1 un algorithme stabilisant vers une spécification P_1.
- Soit S_2 un algorithme stabilisant vers une spécification P_2 si P_1 est vraie.
- Supposons que S_1 ne modifie pas les variables de S_2 une fois que P_1 est vraie.
- Chaque processus exécute alternativement un pas de S_1 et de S_2.

Alors la *composition S de* S_1 *et de* S_2 notée $S_1 \triangleleft S_2$ stabilise vers P_2

Application du théorème

Corollaire
Il existe un algorithme auto-stabilisant qui effectue l'exclusion mutuelle dans un anneau anonyme.

Démonstration.

- Faire la composition des deux algorithmes $S_1 \triangleleft S_2$
  - S_1 : correspond à l'algo. auto. orientant l'anneau.
  - S_2 : correspond à l'algo. auto. de circulation de jeton dans un anneau orienté.
- S_1 et S_2 satisfont les hypothèses du précédent théorème.

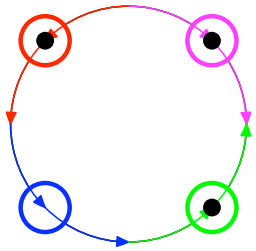
□

Orientation d'un anneau uniforme.

- Objectif : orienter les arêtes de l'anneau de telle façon que l'anneau devient un anneau orienté.
- Hypothèse :
  - même code sur tous les processus.
  - anneau anonyme.
  - chaque processus possède deux liens :`prec` et `succ`.
  - Le mode de communication : par registres.
  - Le démon : centralisé et équitable

Idée de l'algorithme

- les processus envoient des jetons en fonction de leur orientation
- lors de rencontre de deux jetons, ils disparaissent en modifiant les orientations.



Description des variables et des états de l'algorithme

Variables locales

S : Send (prêt à émettre le jeton)

R : Receive (prêt à recevoir le jeton)

I : Idle

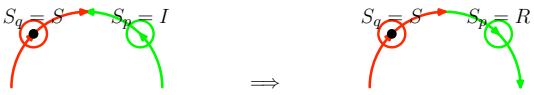
Un processus p tient un jeton

1. S'il est dans l'état S.
2. S'il est dans l'état R et que son prédécesseur :
   - 2.1 n'est pas dans l'état S.
   - 2.2 est dans l'état S mais $succ(prec(\mathtt{p})) \neq p$.

Différentes règles : règle (1)

$S_p \in \{S, R, I\}$

1. $\{S_p = I \wedge S_q = S \wedge lqp = preq\} \rightarrow$
   $S_p \leftarrow R$; si $lpq = succ$ alors échanger succ et pred.



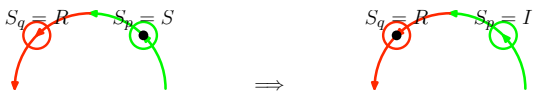
Actions	q	p	p
(1)	$\overrightarrow{S}$	I	$\overrightarrow{R}$

p accepte le jeton et que q soit son prédécesseur

Différentes règles : règle (2)

$S_p \in \{S, R, I\}$

2. $\{S_p = S \wedge lpq = succ \wedge S_q = R \wedge lqp = preq\} \rightarrow$
   $S_p \leftarrow I$; *transmission de jeton*



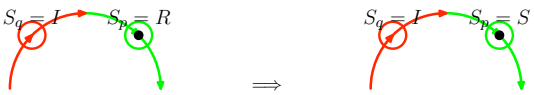
Actions	q	p	p
(2)	$\overleftarrow{R}$	$\overleftarrow{S}$	$\overleftarrow{I}$

p constate que q est son successeur et accepte son jeton.

Différentes règles : règle (3)

$S_p \in \{S, R, I\}$

3. $\{S_p = R \wedge lpq = pred \wedge \neg(S_q = S \wedge lqp = succ)\} \rightarrow$
   $S_p \leftarrow S$;



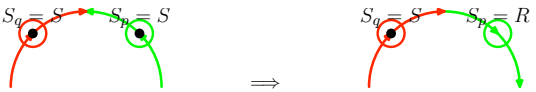
Actions	q	p	p
(3)	$\neg\overrightarrow{S}$	$\overrightarrow{R}$	$\overrightarrow{S}$

p constate que q devient `Iddle` et il se prépare à émettre un jeton.

Différentes règles : règle (4)

$S_p \in \{S, R, I\}$

4. $\{S_p = S_q = S \wedge lpq = lqp = succ\} \rightarrow$
   $S_p \leftarrow R$; échanger succ et pred.



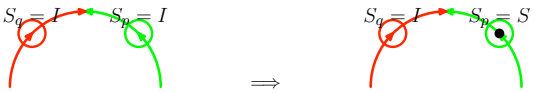
Actions	q	p	p
(4)	$\overrightarrow{S}$	$\overleftarrow{S}$	$\overrightarrow{R}$

p accepte le jeton de q et détruit son jeton

Différentes règles : règle (5)

$S_p \in \{S, R, I\}$

5. $\{S_p = S_q = I \wedge lpq = lqp = succ\} \rightarrow$
   $S_p \leftarrow S$;



Actions	q	p	p
(5)	$\overrightarrow{I}$	$\overleftarrow{I}$	$\overleftarrow{S}$

p observe un problème d'orientation et il crée un jeton.

Récapitulatif de l'algorithme

$S_p \in \{S, R, I\}$

1. $\{S_p = I \wedge S_q = S \wedge lqp = preq\} \rightarrow$
   $S_p \leftarrow R$; si $lpq = succ$ alors échanger succ et pred.
2. $\{S_p = S \wedge lpq = succ \wedge S_q = R \wedge lqp = preq\} \rightarrow$
   $S_p \leftarrow I$; *transmission de jeton*
3. $\{S_p = R \wedge lpq = pred \wedge \neg(S_q = S \wedge lqp = succ)\} \rightarrow$
   $S_p \leftarrow S$;
4. $\{S_p = S_q = S \wedge lpq = lqp = succ\} \rightarrow$
   $S_p \leftarrow R$; échanger succ et pred.
5. $\{S_p = S_q = I \wedge lpq = lqp = succ\} \rightarrow$
   $S_p \leftarrow S$;

Etude de l'algorithme

Actions	q	p	p
(1)	$\overrightarrow{S}$	I	$\overrightarrow{R}$
(2)	$\overleftarrow{R}$	$\overleftarrow{S}$	$\overleftarrow{I}$
(3)	$\neg\overrightarrow{S}$	$\overrightarrow{R}$	$\overrightarrow{S}$
(4)	$\overrightarrow{S}$	$\overleftarrow{S}$	$\overrightarrow{R}$
(5)	$\overrightarrow{I}$	$\overleftarrow{I}$	$\overleftarrow{S}$

Enchaînement des actions

- Après (1) : (3)
- Après (2) : (1) ou (5)
- Après (3) : (1) ou (4)
- Après (4) : (3)
- Après (5) : (2) ou (4)

Mouvement des jetons

Actions	q	p	$\rightarrow$	q	p
(1)	•	×	$\rightarrow$	•	×
(2)	×	•	$\rightarrow$	•	×
(3)	??	•	$\rightarrow$	=	•
(4)	•	•	$\rightarrow$	•	×
(5)	×	×	$\rightarrow$	×	•

Etude de l'algorithme

Actions	q	p	p
(1)	$\overrightarrow{S}$	I	$\overrightarrow{R}$
(2)	$\overleftarrow{R}$	$\overleftarrow{S}$	$\overleftarrow{I}$
(3)	$\neg\overrightarrow{S}$	$\overrightarrow{R}$	$\overrightarrow{S}$
(4)	$\overrightarrow{S}$	$\overleftarrow{S}$	$\overrightarrow{R}$
(5)	$\overrightarrow{I}$	$\overleftarrow{I}$	$\overleftarrow{S}$

Enchaînement des actions

- Après (1) : (3)
- Après (2) : (1) ou (5)
- Après (3) : (1) ou (4)
- Après (4) : (3)
- Après (5) : (2) ou (4)

Mouvement des jetons

Actions	q	p	$\rightarrow$	q	p
(1)	•	×	$\rightarrow$	•	×
(2)	×	•	$\rightarrow$	•	×
(3)	??	•	$\rightarrow$	=	•
(4)	•	•	$\rightarrow$	•	×
(5)	×	×	$\rightarrow$	×	•

49/55 49

Etude de l'algorithme

Actions	q	p	p
(1)	$\overrightarrow{S}$	I	$\overrightarrow{R}$
(2)	$\overleftarrow{R}$	$\overleftarrow{S}$	$\overleftarrow{I}$
(3)	$\neg\overrightarrow{S}$	$\overrightarrow{R}$	$\overrightarrow{S}$
(4)	$\overrightarrow{S}$	$\overleftarrow{S}$	$\overrightarrow{R}$
(5)	$\overrightarrow{I}$	$\overleftarrow{I}$	$\overleftarrow{S}$

Enchaînement des actions

- Après (1) : (3)
- Après (2) : (1) ou (5)
- Après (3) : (1) ou (4)
- Après (4) : (3)
- Après (5) : (2) ou (4)

Mouvement des jetons

Actions	q	p	$\rightarrow$	q	p
(1)	•	×	→	•	×
(2)	×	•	→	•	×
(3)	??	•	→	=	•
(4)	•	•	→	•	×
(5)	×	×	→	×	•

Etude de l'algorithme

Actions	q	p	p
(1)	$\overrightarrow{S}$	I	$\overrightarrow{R}$
(2)	$\overleftarrow{R}$	$\overleftarrow{S}$	$\overleftarrow{I}$
(3)	$\neg\overrightarrow{S}$	$\overrightarrow{R}$	$\overrightarrow{S}$
(4)	$\overrightarrow{S}$	$\overleftarrow{S}$	$\overrightarrow{R}$
(5)	$\overrightarrow{I}$	$\overleftarrow{I}$	$\overleftarrow{S}$

Enchaînement des actions

- Après (1) : (3)
- Après (2) : (1) ou (5)
- Après (3) : (1) ou (4)
- Après (4) : (3)
- Après (5) : (2) ou (4)

Mouvement des jetons

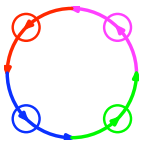
Actions	q	p	$\rightarrow$	q	p
(1)	•	×	→	•	×
(2)	×	•	→	•	×
(3)	??	•	→	=	•
(4)	•	•	→	•	×
(5)	×	×	→	×	•

Preuve (1/4)

A prouver la post-condition ϕ :

pour tout processeur p et q, $pred(p) = q \leftrightarrow succ(q) = p$

Soit L des configurations légitimes (configuration où tous les processeurs pointent dans la même direction).



Preuve (2/4)

Soit L des configurations légitimes.

Lemme (propriété de correction du système)

1. Si $\sigma \in L$ alors $\phi(\sigma)$
2. Si $\sigma \in L$ et $\sigma \rightsquigarrow \sigma'$ alors $\sigma' \in L$

Démonstration.

- Le point (1) est prouvé par définition.
- Seules les règles (1), (2) et (3) sont applicables et elles ne modifient pas les orientations

□

Preuve (3/4)

Lemme

Une configuration terminale (aucune règle à déclencher) est légitime.

Démonstration.

- Soit σ une configuration terminale.
- aucun processus n'est dans R (sinon application de (2) ou (3))
- Par contradiction, supposons que σ n'est pas orienté :

  Cela signifie que 2 processus pointent l'un vers l'autre.

  - Si *un* des *deux* sont dans I,
  - sinon les deux sont dans S : application de (4)

□

Preuve (3/4)

Lemme

Une configuration terminale (aucune règle à déclencher) est légitime.

Démonstration.

- Soit σ une configuration terminale.
- aucun processus n'est dans R (sinon application de (2) ou (3))
- Par contradiction, supposons que σ n'est pas orienté :

  Cela signifie que 2 processus pointent l'un vers l'autre.

  - Si *un* des *deux* sont dans I,
    1. si l'autre dans S, alors application de (1)
    2. si l'autre dans I, alors application de (5)
  - sinon les deux sont dans S : application de (4)

□

Preuve (3/4)

Lemme
Une configuration terminale (aucune règle à déclencher) est légitime.

Démonstration.

- Soit σ une configuration terminale.
- aucun processus n'est dans R (sinon application de (2) ou (3))
- Par contradiction, supposons que σ n'est pas orienté :

  Cela signifie que 2 processus pointent l'un vers l'autre.
  - Si *un* des *deux* sont dans I, application de (1) ou (5)
  - sinon les deux sont dans S : application de (4)

□

Preuve (4/4)

Théorème
L'algorithme converge vers les états légitimes.

Démonstration.

- Considérons la règle 5.
  - c'est la seule règle qui crée un jeton
  - les autres règles ne peuvent pas déclencher la règle 5.
  - Son exécution est déclenchée par une mauvaise initialisation.
- Donc le nombre de jeton est bornée par n le nbre de processus.
- Si un jeton parcourt une distance n, alors il est orienté.
- Le nombre de pas où le jeton se déplace sans qu'une configuration soit atteinte est au plus $n(n-1)$

□

Plan

Récapitulatif

- Soit $S = (C, \rightarrow)$ un système distribué où
  C : ens des configurations et $\rightarrow$: relation de transition.
- Soit $L \subseteq C$: ensemble de configurations légitimes

Un algorithme est *auto-stabilisant* si

1. à partir toutes les configurations possibles γ, son exécution converge vers une configuration légitime.
2. à partir toutes les configurations légitimes γ, son exécution est composée uniquement de configurations légitimes.