Algorithmes distribuées auto-stabilisants.

Exercice : distance dans un graphe.

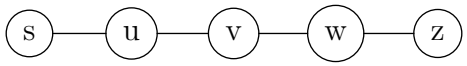
Soit le graphe $G = (V, E)$ (V correspond à l'ensemble des sommets et E l'ensemble des arêtes) et u un sommet de G. Pour chaque sommet v de G, la variable locale $t(v)$ correspondra un entier. Par convention, l'ensemble des voisins du sommet v sera noté $\Gamma(v)$. Le modèle considéré est le suivant :

1. le démon est équitable et centralisé.
2. le modèle de communication est celui des communications par mémoire partagée

Considérons l'algorithme suivant :

- `pour le sommet` u : $true \rightarrow t(u) := 0$
- `pour le sommet` $v \neq u$: $true \rightarrow t(v) := \min_{j \in \Gamma(v)}\{t(j) + 1\}$

Question 1 : En considérant la topologie ci-dessus, la configuration du système est la suivante ($\texttt{t(s)} = 3, \texttt{t(u)} = 0, \texttt{t(v)} = 8, \texttt{t(w)} = 5, \texttt{t(z)} = 3$) Supposons que le démon active d'abord, le noeud `s`, puis `w`, puis `z`, puis `w`, puis `v`, puis `s`, puis `w`. A chaque activation du démon, donner la configuration du système.



Correction :

CONFIGURATION DE DEPART.

($\texttt{t(s)} = 3$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 8$ $\texttt{t(w)} = 5$ $\texttt{t(z)} = 3$)

le démon active le noeud `s`,

($\texttt{t(s)} = 1$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 8$ $\texttt{t(w)} = 5$ $\texttt{t(z)} = 3$)

le démon active le noeud `w`,

($\texttt{t(s)} = 1$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 8$ $\texttt{t(w)} = 4$ $\texttt{t(z)} = 3$)

le démon active le noeud `z`,

($\texttt{t(s)} = 1$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 8$ $\texttt{t(w)} = 4$ $\texttt{t(z)} = 5$)

le démon active le noeud `w`,

($\texttt{t(s)} = 1$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 8$ $\texttt{t(w)} = 6$ $\texttt{t(z)} = 5$)

le démon active le noeud `v`,

($\texttt{t(s)} = 1$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 1$ $\texttt{t(w)} = 6$ $\texttt{t(z)} = 5$)

le démon active le noeud `s`,

($\texttt{t(s)} = 1$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 1$ $\texttt{t(w)} = 6$ $\texttt{t(z)} = 5$)

le démon active le noeud `w`,

($\texttt{t(s)} = 1$ $\texttt{t(u)} = 0$ $\texttt{t(v)} = 1$ $\texttt{t(w)} = 2$ $\texttt{t(z)} = 5$)

Question 2 : Que calcule l'algorithme ? Caractériser les configurations légitimes dans notre contexte.
Correction :

1. l'algorithme calcule le plus court chemin entre u et tous les autres sommets.

2. Soit $d(u,v)$ la distance entre v et u dans G. Soit $V(G)$ l'ensemble de sommets de G.

$$\text{configuration légitime} = \{t(v) = d(v,u), \forall v \in V(G)\}$$

Question 3 : Prouver que votre algorithme satisfait la propriété suivante : toute exécution commencant par une configuration légitime est composée uniquement de configurations légitimes.
Correction :

Soit L une configuration légitime. Soit v le sommet de G qui est activé par le démon G.

- Si $v = u$: alors l'action "$true \rightarrow t(u) := 0$" est activée.
  Donc $t(u) = 0 = d(u,u)$
  Donc la nouvelle configuration reste légitime (en fait, l'action ne modifie pas la configuration du système).
- Si $v \neq u$: alors l'action $true \rightarrow t(v) := \min_{j \in \Gamma(v)}\{t(j)+1\}$ est activée.
  Comme L est légitime, alors $\forall j \in \Gamma(v)$, on a $t(j) = d(u,j)$. Après l'exécution de cette action,
  $t(v) = \min_{j \in \Gamma(v)}\{t(j)+1\}$
  $t(v) = \min_{j \in \Gamma(v)}\{d(j,u)+1\}$
  $t(v) = d(v,u)$
  Donc la nouvelle configuration reste légitime (en fait, l'action ne modifie pas la configuration du système).

Exercice : composition

Dans cet exercice, nous considérons deux algorithmes stabilisant S_1 et S_2 tel que

1. aucune variable modifiée par S_2 soit utilisée par S_1
2. S_1 stabilise vers une spécification θ.
3. S_2 stabilise vers une spécification ϕ sachant que θ est satisfaite.
4. S_1 ne change pas de variables lues par S_2 une fois que la spécification θ est satisfaite

A partir de S_1 et de S_2, nous allons construire un algorithme S de la façon suivante :

1. S possède toutes les variables et toutes les actions (instructions) de S_1 et de S_2.
2. chaque processus exécute alternativement une action de S_1 et une de S_2.

Pour une configuration γ de S, $\gamma^{(i)}$ est la projection de γ composée des variables de S_i. La notion de projection est étendue pour les exécutions. Pour une exécution $E = (\gamma_0, \gamma_1, \ldots)$ de S, $E^{(i)}$ est la séquence $(\gamma_0^{(i)}, \gamma_1^{(i)}, \ldots)$. Nous considérons par la suite une exécution infinie $E = (\gamma_0, \gamma_1, \ldots)$ de S sachant que le démon est centralisé et équitable.

Question 1 : Montrer que $E^{(1)}$ est une exécution de S_1.
Correction :

Comme S_2 ne modifie aucune variable utilisé par S_1, alors $E^{(1)}$ est composée par une séquence de configurations telle que $\gamma_1^{(i)}$ se transforme en $\gamma_1^{(i+1)}$ par le fait que seul S_1 modifie les variables de $\gamma_1^{(i)}$.

Donc $E^{(1)}$ est une exécution de S_1.

Question 2 : Que pouvez-vous déduire de $E^{(1)}$?
Correction :

Comme le démon est équitable, S_1 stabilise vers une spécification θ. Il existe une configuration $\gamma_i^{(1)}$ de $E^{(1)}$ à partir de laquelle toutes les configurations suivantes satisfont la spécification θ.

Question 3 : Montrer que S stabilise vers une spécification ϕ.
Correction : Considérons $E' = (\gamma_i, \gamma_{i+1}, \ldots)$ une sous-exécution de E telle que $\gamma_i^{(1)}$ satisfait la spécification θ.

Considérons l'exécution $E'^{(2)} = (\gamma_i^{(2)}, \gamma_{i+1}^{(2)}, \ldots)$.

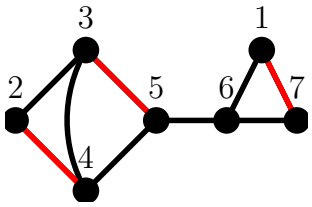
Comme S_1 ne change pas de variables lues par S_2 une fois que la spécification θ est satisfaite, alors $E'^{(2)}$ est une exécution de S_2. Alors il existe une configuration $\gamma_\ell^{(2)}$ de $E'^{(2)}$ à partir de laquelle toutes les configurations suivantes satisfont la spécification ϕ.

Donc il existe une configuration γ_ℓ de E à partir de laquelle toutes les configurations suivantes satisfont la spécification ϕ.

Ainsi S stabilise vers une spécification ϕ.

Exercice : couplage maximal dans un graphe.

Un `couplage` d'un graphe est un ensemble d'arêtes non-adjacentes, c'est-à-dire telles que chaque sommet du graphe appartient à au plus une arête (voir figure 2). Il est `maximal` s'il ne peut être augmenté comme celui représenté en figure 1.



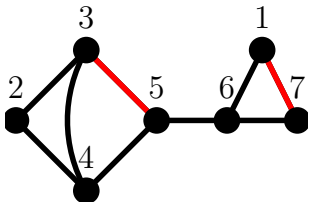


Figure 1 et Figure 2
Les arêtes rouges forment un couplage maximal du graphe.

Question 1 : Trouver un autre couplage maximal du graphe de la figure 1.
Correction : Un couplage maximal est $\{(1,7),(5,6),(3,4)\}$

Le modèle à lecture d'état est considéré. Chaque processus p possède une variable locale `pref(`p`)` dont la valeur est $\emptyset \cup \Gamma(p)$ avec $\Gamma(p)$ qui est l'ensemble des voisins de p. La valeur `pref(`p`)` désigne avec quel voisin il est associé dans le couplage.

Question 2 : Calculer la valeur `pref` des sommets du graphe de la figure 1.
Correction :

- `pref(1)=7`
- `pref(2)=4`
- `pref(3)=5`
- `pref(4)=2`
- `pref(5)=3`
- `pref(6)=`$\emptyset$
- `pref(7)=1`

Nousnous allons définir un état pour chaque processus p.

- Si le processus p a choisi le processus q alors
  - p `attend` si q n'a pas encore choisi

  - p est `apparié` si q a choisi p
  - p est `chaînant` si q a choisi un autre que p
- Si le processus p n'a pas choisi le processus q alors
  - p est `mort` si tous ses vousins sont appariés q n'a pas encore choisi
  - p est `libre` dans le cas contraire

Question 3 : Calculer les états des sommets du graphe G représenté en figure 2.
Correction :

- `etat(1)=7`
- `etat(2)=libre`
- `etat(3)=apparié`
- `etat(4)=libre`
- `etat(5)=apparié`
- `etat(6)=mort`
- `etat(7)=1`

Considérons la spécification ϕ suivante pour une configuration γ :
$\phi = \forall p\ (etat(p) = \texttt{apparié} \vee etat(p) = \texttt{mort})$

Question 4 : Montrer que pour une configuration γ si $\phi(\gamma)$ est vraie alors l'ensemble des arêtes $M = \{(p, pref(p)) : pref(p) \neq \emptyset\}$ est un couplage maximal.
Correction : voir le document envoyé

Considérons l'algorithme suivant :
$\texttt{M}_\texttt{p}$: $\{pref(p) = \emptyset \wedge pref(q) = p\} \rightarrow pref(p) := q$
$\texttt{S}_\texttt{p}$: $\{pref(p) = \emptyset \wedge (\forall r \in \Gamma(p), pref(r) \neq p) \wedge pref(q) = \emptyset\} \rightarrow pref(p) := q$
$\texttt{U}_\texttt{p}$: $\{pref(p) = q \wedge pref(q) \neq p \wedge pref(q) = \emptyset\} \rightarrow pref(p) := \emptyset$

Question 5 : Montrer que la configuration γ est terminale[1] si et seulement si $\phi(\gamma)$ est vraie.
Correction : voir le document envoyé

Question 6 : L'algorithme atteint une configuration terminale en plus en $O(n^2)$ pas avec n le nombre de processus. *Indication :* Etudier la fonction F définit de la façon suivante : soit γ une configuration possible $F(\gamma) = (c + f = w, 2c + f)$ *avec*

- c le nombre de processus chaînants
- f le nombre de processus libres
- w le nombre de processus en attente

Correction : voir le document envoyé

Exercice : élection de leader dans un réseau en général

On suppose ici que chaque processeur a une identité unique allant de 1 à N où N est un majorant du nombre de processeurs noté n (i.e $N \geq n$). L'objectif de cet exercice est d'élire un processeur parmi tous et de propager le résultat de cette élection.

Considérons l'algorithme suivant décrit de façon informelle. Chaque processeur P_i est candidat pour être le leader. Au début, il se désigne comme tel. De façon itérative, P_i communique à ces voisins son identité x du leader. Quand P_i reçoit une identité y d'un autre candidat par ces voisins, si $x > y$, alors P_i choisit y comme son identité du nouveau leader.

[1] aucune commande dans cette configureation peut être déclenchée

Question 1 : Cet algorithme est-il auto-stabilisant ?
Correction : Non. Il existe des configurations telle qu'un processeur décide que z est le leader
- tel que la valeur de z est la plus petite que toutes les autres identités
- tel que la valeur ne correspond à aucune identité des processeurs.

L'algorithme va converger vers une configuration où tous les processeurs vont élire z ne correspondant à aucun processeur du système.

Maintenant considérons l'algorithme suivant en supposant que chaque processeur P_i connaît la valeur N. De plus, le processeur P_i définit le leader en fonction de deux variables
- $leader_i$ l'identité du leader
- $distance_i$ la valeur de distance entre lui et le leader.

De façon itérative, P_i lit les variables liées au leader de ces voisins. En fonction de cette lecture, il choisit le leader ayant la plus petite identité et tel que la distance entre lui et le leader soit inférieur à N.

Voici sa description formelle pour le noeud courant P_i :
$true \rightarrow$

1. $\langle candidat, distance\rangle := \langle ID, 0\rangle$
2. pour chaque $P_j \in \Gamma(P_i)$ faire
   (a) $\langle leader[j], distance[j]\rangle := lire(\langle leader_j, distance_j\rangle$
   (b) Si $(distance[j] < N)$ et si $(leader[j] < candidat)$ alors
   $\langle candidat, distance\rangle := \langle leader[j], distance[j] + 1\rangle$
3. écrire $\langle leader_j, distance_j\rangle = \langle candidat, distance\rangle$

Question 2 Prouver que pour chaque exécution équitable qui commence à partir de n'importe quelle configuration, il existe une configuration c à partir de laquelle tous les candidats leaders de chaque processeur correspondent bien une identité d'un processeur.
Correction : Pour prouver cette assertion, il suffit de prouver que tant qu'il existe des identités qui ne correspondent à aucun processeur alors, la minimal distance entre "le virtuel processeur possédant cette identité" et les noeuds le choisissant augmente. (A un moment donné, la distance sera plus grande que N)

Soit $\mathcal{P}(c)$ un ensemble des processeurs qui choisissent une identité d'un processeur virtuel comme leader dans la configuration c. Soit P_i apartenant à $\mathcal{P}$ tel que sa distance entre lui et le processeur virtuel soit la plus petite possible. Comme nous nous considérons dans une exécution équitable, il existe une transition entre c_1 et c_2 où P_i change son état. P_i change son état en se choisissant tout d'abord lui comme leader. Puis il compare son choix à tous ses autres voisins.

1. S'il choisit une identité attribuée à un processeur, alors $\mathcal{P}(c_1) > \mathcal{P}(c_2)$. Le nombre de processeurs choisissant une identité virtuelle diminue.
2. Si il choisit une identité attribuée à un processeur "virtuel", alors sa distance entre lui et le processeur virtuel augmente au pire de 1.

Question 3 Prouver que pour chaque exécution équitable qui commence à partir de n'importe quelle configuration c, il existe une configuration c à partir de laquelle toutes les configurations suivantes satisfont le prédicat suivant :

`P` : tous les processeurs sélectionnent le processeur ayant la plus petite identité comme leader.

Correction : Il suffit de prouver qu'il existe une suite de configuration $c_0, c_1, c_2, \ldots, c_i$ dans chaque exécution équitable qui commence à partir de n'importe quelle configuration c tel que
- c_i précède c_{i+1}
- c_i satisfait le prédicat suivant

`P_i` : tous les processeurs à distance i du leader sélectionnent le processeur ayant la plus petite identité comme leader.

Tout d'abord, il faut considérer la configuration c' succédant à c dans laquelle tous les candidats leaders de chaque processeur correspondent bien un identifieur d'un processeur. D'après la question 2, une telle configuration existe. On peut prouver cela par récurrence sur i.

- cas de base $i = 0$: Comme l'exécution est équitable, il existe une configuration où le processeur leader est activé. Il change son état de telle façon qu'il décide d'être le leader et qu'il est à distance 0.
- cas d'induction i : Par hypothèse d'induction, il existe une configuration c_{i-1} tel que tous les processeurs à distance i du leader sélectionnent le processeur ayant la plus petite identité comme leader.
  Comme l'exécution est équitable, il existe une configuration c'_i succédant à c_{i-1} où un processeur à distance i du leader est activé. Il change son état de telle façon qu'il décide d'être le leader et qu'il est à distance i. Cela se déroulera de la même façon pour tous les processeurs à distance i.