Université de Strasbourg
UFR Mathématique et Informatique

M1 Informatique
2019/2020

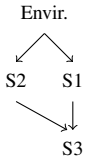
Algorithmes Distribués

TD n° 4 : Terminaison

Exercice 1 : Algorithme de détection de terminaison de Dijkstra-Scholten

Le code de l'algorithme de calcul diffusant de Dijkstra et Scholten est rappelé en annexe 2.

Q 1. Compléter le tableau des valeurs des variables père, defin, defout des processus env, S1, S2 et S3 dans le scénario ci-dessous :



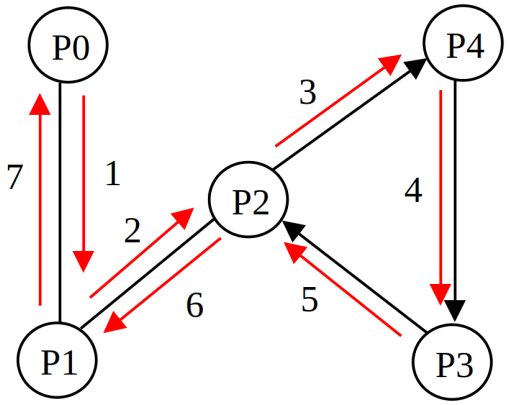
		env	S1	S2	S3
	val. init.	- 0 0	- 0 0	- 0 0	- 0 0
	env. active s1	...	...	...	...
	s1 active s3	...	...	...	...
pas de changement, car $aval_1 \neq 0$	calcul s1 terminé	...	...	...	...
	env. active s2	...	...	...	...
	s2 active s3	...	...	...	...
s3 envoie signal à s2	calcul s3 terminé	...	...	...	...
s2 ne propage pas (non terminé)		...	...	...	...
s3 envoie signal à s1	calcul s3 terminé	...	...	...	...
s1 envoie signal à e (terminé)		...	...	...	...
S2 envoie signal à e	calcul s2 terminé	...	...	...	...

Q 2. Montrer que lorsque le calcul diffusant se termine, alors le processus environnement revient à son état initial (defin = defout = 0).

Exercice 2 : Algorithme de détection de terminaison de Misra

Le code de l'algorithme de détection de terminaison sur un réseau quelconque de Misra est fourni en Annexe 3.

Q 1. Appliquer l'algorithme de Misra au scénario suivant :



Initialement P0, P1, P2 sont passifs et P3, P4 sont actifs.

1. P0 lance une détection de terminaison
   couleur ← .....; nb ← .....; jeton_présent ← .....;
   envoi (jeton, 0) à .....
2. P3 envoie un message à P2
3. ..... reçoit (jeton, 0) de P0
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← .....; (car couleur = noir)
   envoi (jeton, ...) à .....
   couleur ← .....; jeton_présent ← .....;
4. ..... reçoit (jeton, ...)
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← .....;
   envoi (jeton, ...) à .....
   couleur ← .....; jeton_présent ← .....;
5. P2 reçoit le message envoyé par P3
   etat ← .....; couleur ← .....;
6. ..... reçoit (jeton, ...)
   jeton_présent ← .....
   ..... attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← .....;
   envoi (jeton, ...) à .....
   couleur ← .....; jeton_présent ← .....;
7. ..... reçoit (jeton, ...)
   jeton_présent ←
   Le processus attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← .....;
   envoi (jeton, ...) à .....
   couleur ← .....; jeton_présent ← .....;
8. ..... reçoit (jeton, ...)
   jeton_présent ← vrai
   Le processus attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← .....;
   envoi (jeton, ...) à .....
   couleur ← .....; jeton_présent ← .....;
9. ..... reçoit (jeton, ...)
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← .....;
   envoi (jeton, ...) à ..... couleur ← .....; jeton_présent ← .....;
10. ..... reçoit (jeton, ...)
    jeton_présent ← vrai
    Comme état = passif on exécute l'émission du jeton
    nb ← .....;
    envoi (jeton, ...) à .....
    couleur ← .....; jeton_présent ← .....;
11. ..... reçoit (jeton, ...)
    jeton_présent ← vrai

Comme état = passif on exécute l'émission du jeton
nb ← ..... ;
envoi (jeton, ...) à .....
couleur ← ..... ; jeton_présent ← ..... ;

12. l'algorithme continue de la même manière (tous les processus sont passifs)
- ..... reçoit (jeton, ...) et envoie (jeton, ...) à .....
- ..... reçoit (jeton, ...) et envoie (jeton, ...) à .....
- ..... reçoit (jeton, ...) et envoie (jeton, ...) à .....
- ..... reçoit (jeton, ...) et envoie (jeton, ...) à .....
- ..... reçoit (jeton, ...) = taille (C) et couleur = blanc => terminaison détectée

Annexe 2 : algorithme de détection de terminaison de Dikkstra et Scholten

Pour chaque processus Pi deux variables entières sont définies, defin et defout. Elles comptent respectivement la somme des déficits des arcs entrant vers le processus et la somme des déficits des arcs sortant. Initialement ces deux compteurs sont nuls.
Pi gère ses compteurs defin et defout de la façon suivante :
• envoi d'un message : defout ← defout + 1 ;
• réception d'un message : defin ← defin + 1 ;
• envoi d'un signal : defin ← defin – 1 ;
• réception d'un signal : defout ← defout – 1 ;

Algorithme
Outre les compteurs defin et defout, le processus Pi est doté de deux variables :
• père : qui représente son père dans l'arbre de recouvrement du graphe des processus (celui-ci est déterminé par la première communication qu'a le processus)
• appelants : ensemble dans lequel le processus mémorise les noms des processus autres que le père qui lui ont envoyé un message

var : defin, defout : 0 .. nbmax initialisé à 0 ;
père : 1 .. n ;
appelants : sac* de 1 .. n ;

Début
lors de réception de (message, expd) de expd ;
début
  si defin = 0 **alors**
    père ← expd ;
  sinon
    appelants appelants ⊕ expd ;
  fin
  defin ← defin + 1 ;
fin

lors de réception de (signal, expd) de expd ;
début
  defout ← defout – 1 ;
fin

lors de désir d'émettre (message, i) vers j ;
début
  si defin ≠ 0 **alors**
    defout ← defout + 1 ;
    envoyer (message, i) à j ;
  fin
fin

```
lors de désir d'émettre (signal) ;
début
    si (defin > 1) ou (defin = 1 et defout = 0) alors
        si defin = 1 alors
            /* envoi du dernier signal : defout = 0 */ envoyer (signal, i) à père ;
        sinon
            aux ← un élément de appelants ;
            appelants ← appelants \ {aux} ;
            envoyer (signal, i) à aux ;
        fin
        defin ← defin – 1 ;
    fin
fin
```

Annexe 3 : algorithme de détection de terminaison de Misra sur un réseau de topologie quelconque

Hypothèses :

• pas de perte des messages

• pas de déséquencement (canaux FIFO)

On suppose que sur le circuit C (pré-calculé) les deux fonctions suivantes sont définies :

• fonction taille (C : circuit) résultat entier : donne la taille du circuit

• fonction successeur (C : circuit, i : 1 .. n) résultat 1 .. n : donne pour le processus Pi qui l'exécute, le numéro du successeur de Pi sur le circuit.

Chaque processus Pi est doté des déclarations suivantes :

var couleur : (blanc, noir) initialisé à noir ;
état : (actif, passif) initialisé à actif ;
$jeton_{prsent}$: booléen initialisé à faux ;
nb : entier initialisé à 0 ;

La variable couleur est associée au processus et nb sert à mémoriser la valeur associée au jeton entre sa réception et sa réémission. Les messages sont de deux types : messages, jeton.

Algorithme :

```
lors de réception de (message, m) ;
début
    état ← actif ;
    couleur ← noir ;
fin

lors de attente (message, m) ;
début
    état ←passif;
fin

lors de réception de (jeton, j) ;
début
    nb ←j ;
    jeton_prsent ←vrai;
    si nb = taille (C) et couleur = blanc alors
        terminaison détectée ;
    fin
fin
```

```
lors de émission de (jeton, j) ;
début
    si jeton_prsent et état = passif alors
        si couleur = noir alors
            nb ← 0;
        fin
        sinon
            nb ← nb + 1;
        fin
        envoyer (jeton, nb) à successeur (C, i) ;
        couleur ← blanc ;
        jeton_prsent ← faux ;
    fin
fin
```