




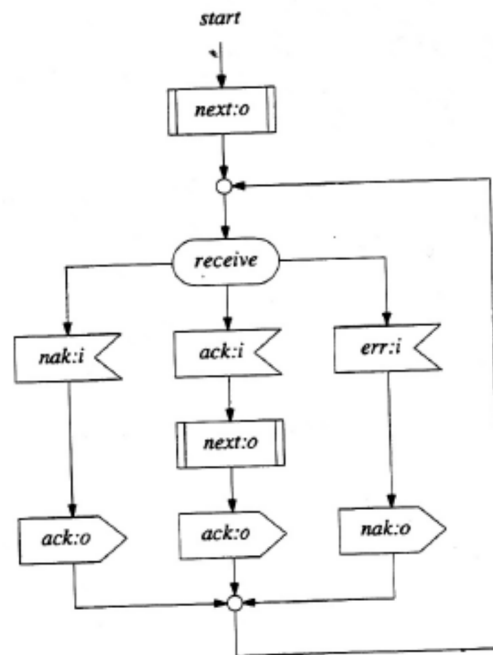
Algorithmes Distribués

 Promela

Exercice 1 : Protocole de Lynch

Ce protocole est réalisé par deux processus distants A et B qui communiquent par échange de messages.

Q 1. Compréhension du fonctionnement du protocole décrit par l'automate ci-dessous :



1. le transfert de données peut-il avoir lieu dans un seul sens à la fois ?
2. l'échange est-il « half-duplex » (bidirectionnel à l'alternat) ou « full-duplex » (bidirectionnel simultané) ?
3. que signifient les variables « o » et « i » ?
4. le mécanisme de « piggy-backing » est-il utilisé ?
5. quand est-ce que l'émetteur de données produit un nouveau message à envoyer ?
6. quand est-ce que le récepteur de données consomme un nouveau message ?
7. comment peut-on initialiser le transfert de données ?
8. la fin du transfert de données est-elle prévue dans le protocole ?
9. peut-il y avoir des duplications de messages ? Donnez un exemple de scénario

Q 2. Codage en Promela

1. Définir les différents types de messages
2. Définir le type de processus prototype transfer (chan inc, out, chin, chout) : inc et out représentent les canaux de communication entre le processus et sa couche supérieure (production / consommation de données) chin et chout représentent les canaux de communication entre les processus distants (réception / émission de messages)
3. Définir le processus init qui déclare les canaux et lance les processus de type transfer représentant deux processus A et B, pour le scénario suivant :

- le processus A produit les données '1' et '2'
- le processus B produit les données '3' et '4'
- le processus B initialise le transfert en envoyant le message ERR(0) au processus A

Q 3. Testez votre code en TP

1. testez ce code en simulation en exécutant différents scénarios d'échange ;
2. lancez la vérification. Pourquoi donne-t-elle un résultat incorrect ?
3. la fin du transfert de données est-elle prévue dans le protocole ?
4. corrigez le code pour que la vérification soit correcte ;
5. montrez que si l'initialisation se fait dans les deux sens à la fois (envoi du message err(0)), il peut y avoir des duplications de messages. Donnez un exemple de scénario en faisant une simulation guidée interactive.

Exercice 2 : Algorithme d'élection de Chang et Roberts (variante 2)

Q 1. Ecrire le code en Promela de l'algorithme de Chang et Roberts vu en cours, dont le texte est rappelé au verso. On considère ici un anneau de 4 processus P_1, P_2, P_3 et P_4 reliés par des canaux fromtoij. Chaque processus P_i communique avec sa couche supérieure par l'intermédiaire un canal in_i . Ce canal sert à transmettre le message de candidature du site.

La structure de données est la suivante :

```
#define REPOS 0
#define EN_COURS 1
#define TERMINE 2

mtype = { candidat, election, élu }

short nb_elu; /* compte le nb de processus élus */

proctype electeur (short num; chan inc, rec, em)
{ ....}

init
{chan from1to2 = [1] of {byte, byte};
 chan from2to3 = [1] of {byte, byte};
 chan from3to4 = [1] of {byte,byte};
 chan from4to1 = [1] of {byte,byte};
 chan in1 = [1] of {byte};
 chan in2 = [1] of {byte};
 chan in3 = [1] of {byte};
 chan in4 = [1] of {byte};

atomic {
run electeur(1, in1, from4to1, from1to2 );
run electeur(2, in2, from1to2, from2to3 );
run electeur(3, in3, from2to3, from3to4 );
run electeur(4, in4, from3to4, from4to1 );

in1 ! candidat; in2 ! candidat; in3 ! candidat; in4 ! candidat
}
}

/* dans ce scénario tous les processus sont candidats */
```

Q 2. Rajoutez l'assertion permettant de vérifier qu'un seul processus est élu.

Q 3. Rajoutez une variable seuil permettant de modéliser la capacité d'un site (par rapport à sa charge) : au dessus de cette valeur le site sera déclaré comme inapte à l'élection car sa charge courante considérée par le système comme trop élevée.

Q 4. Testez votre code en TP.

I Annexes

Rappel de l'algorithme de Chang et Roberts (variante 2) :

Variables du processus P_i :

état : état du service (repos, en_cours, terminé).
/* Cette variable est initialisée à repos. */
chef : identité du site élu

Algorithme du site i :

Candidature () ;

début

si état = repos **alors**
 état \leftarrow en_cours ;
 chef $\leftarrow i$;
 env_vg (election, i) ;

fin

 attendre (état = termine) ;
 renvoyer (chef) ;

fin

/* envoi à la couche supérieure : primitive du service */

sur_reception_de (election, initiateur);

début

si état = repos ou initiateur > chef **alors**
 état \leftarrow en_cours ;
 chef \leftarrow initiateur ;
 envoyer_vg (election, initiateur) ;

fin

sinon

si (i = initiateur) **alors**
 état \leftarrow termine ;
 envoyer_vg (elu, i) ;

fin

fin

fin

sur_reception_de (elu, initiateur);

début

si (i \neq initiateur) **alors**
 envoyer_vg (elu, initiateur) ;
 état \leftarrow termine ;

fin

fin