

03 jui 02 14:35

prog2.net

Page 1/3

```

/*****
/* Modelisation en reseau de Petri de la solution de la question 8 */
/* du sujet de Juin 2002; */
/* Cette version est plus abstraite que le programme Ada dans le */
/* sens ou les entrees internes ne sont pas modelisees; les clients */
/* en attente restent simplement dans les places correspondant aux */
/* appels des entrees externes (places Sert1 et Sert2) */
/*****

#define N      4
#define ALL_Id 0..N-1

/*****
/* Les places pour modeliser les clients */
/*****
#place Repos      lo(<.0.>)      hi(<.N-1.>)      mk(<.ALL_Id.>)
#place Bidon      lo(<.0.>)      hi(<.N-1.>)      mk(<.ALL_Id.>)
#place Choix      lo(<.0,0.>)    hi(<.N-1,N-1.>)

#place P2         lo(<.0,0.>)    hi(<.N-1,N-1.>)
#place Att2      lo(<.0,0.>)    hi(<.N-1,N-1.>)

#place P1         lo(<.0.>)      hi(<.N-1.>)
#place Att1      lo(<.0.>)      hi(<.N-1.>)

#place End       lo(<.0.>)      hi(<.N-1.>)

#place Call2     lo(<.0,0.>)    hi(<.N-1,N-1.>)
#place Ack2     lo(<.0,0,0.>)  hi(<.N-1,N-1,1.>)

#place Call1     lo(<.0.>)      hi(<.N-1.>)
#place Ack1     lo(<.0,0.>)    hi(<.N-1,1.>)

/*****
/* Les places pour modeliser le serveur */
/*****
#place Sert1     lo(<.0.>)      hi(<.N-1.>)
#place Sert2     lo(<.0,0.>)    hi(<.N-1,N-1.>)

#place Cpt1     lo(<.0.>)      hi(<.N.>)      mk(<.0.>)

/*****
/* transitions de debut et de fin d'activite des clients */
/* un site X choisi aleatoirement (de facon indeterministe) une */
/* valeur de site Y; selon cette valeur, le site X agira en tant que */
/* demandeur d'un rdv particulier (avec Y) ou en tant que serveur */
/* pour une demande d'un site Z avec lui (X) */
/*****
#trans t_init
    in {Repos: <.X.>; Bidon: <.Y.>;}
    out {Choix: <.X, Y.>; Bidon: <.Y.>;}
#endtr

#trans t_end
    in {End: <.X.>;}
    out {Repos: <.X.>;}
#endtr

/*****
/* transitions correspondant au comportement des clients lorsqu'ils */
/* demandent a servir un autre client quelconque; cette branche est */
/* choisies lorsque la valeur Y choisie aleatoirement est egale a X */
/* qui represente l'identite du site */
/*****
#trans t11
    in {Choix: <.X, Y.>;}
    out {P1:<.X.>;}
    gate (X==Y);
#endtr

#trans t12
    in {P1:<.X.>;}
    out {Att1:<.X.>; Call1: <.X.>;}
#endtr

#trans t13
    in {Att1: <.X.>; Ack1: <.X, Z.>;}
    out {End: <.X.>;}
#endtr

```

03 jui 02 14:35

prog2.net

Page 2/3

```

/*****
/* transitions correspondant au comportement des clients lorsqu'ils */
/* veulent un rdv avec un site particulier (cas ou X /= Y) */
/*****
#trans t21
    in {Choix: <.X, Y.>;}
    out {P2:<.X,Y.>;}
    gate (X!=Y);
#endtr

#trans t22
    in {P2:<.X,Y.>;}
    out {Att2:<.X,Y.>; Call2: <.X,Y.>;}
#endtr

#trans t23
    in {Att2: <.X,Y.>; Ack2: <.X, Y, Z.>;}
    out {End: <.X.>;}
#endtr

/*****
/* La partie du serveur qui gere les demandes de clients qui */
/* acceptent de servir un rdv quelconque. L'evitement de */
/* l'interblocage possible (lorsque tous les sites choisissent ce */
/* comportement) est obtenu en gerant un compteur sur le nombre de */
/* site en attente sur ce choix; lorsque la valeur de ce compteur est */
/* egale a N on envoie une reponse negative au client (transition */
/* ts1_refus). S'il n'y pas d'interblocage possible, le client est */
/* mis en attente dans la place Sert1; il en ressortira par la */
/* la transition ts_ack decrite plus bas. */
/*****
#trans ts1
    in {Call1: <.X.>; Cpt1: <.K.>;}
    out {Sert1: <.X.>; Cpt1: <.K+1.>;}
#endtr

#trans ts1_refus
    in {Sert1: <.X.>; Cpt1: <.K.>;}
    out {Ack1: <.X, 0.>; Cpt1: <.K-1.>;}
    gate ( K=N );
#endtr

/*****
/* La partie du serveur qui gere les demandes de clients qui */
/* veulent un rdv avec un site particulier; ces clients sont mis en */
/* attente dans la place Sert2, ils en ressortiront par la transition */
/* ts_ack decrite plus bas. L'evitement de l'interblocage du a une */
/* attente circulaire est evite par l'envoi d'une reponse negative */
/* au site demandeur des que le site voulu est deja lui meme en */
/* attente d'un autre site (transition ts20); */
/*****
#trans ts2
    in {Call2: <.X,Y.>; }
    out {Sert2: <.X,Y.>; }
#endtr

#trans ts2_refus
    in {Sert2: <.X,Y.>; Sert2: <.Y, Z.>;}
    out {Ack2: <.X, Y, 0.>; Sert2: <.Y, Z.>;}
#endtr

/*****
/* Partie du serveur gerant les acceptations de rdv. */
/* Lorsqu'un client X est en attente de rdv quelconque (marque <.X.> */
/* dans la place Sert1) et qu'un client Y veut un rdv avec le site */
/* X (marque <.Y,X.> dans la place Sert2) le serveur envoie une */
/* reponse positive aux deux clients X et Y en déposant une marque */
/* dans les places Ack1 et Ack2 */
/*****
#trans ts_ack
    in {Sert1: <.X.>; Sert2: <.Y, X.>; Cpt1: <.K.>;}
    out {Ack1: <.X, 1.>; Ack2: <.Y, X, 1.>; Cpt1: <.K-1.>;}
#endtr

```

03 jui 02 14:35

prog2.net

Page 3/3

```
/* **** */
/* si l'on verifie ce model (absence d'interblocage) avec l'outil */
/* prod en executant la commande prod prog2.init; ./prog2 -s */
/* (l'option -s permet d'utiliser la technique des "stubborn sets" */
/* qui permet de reduire considerablement la taille du graphe des */
/* marquages accessibles genere en reduisant l'entrelacement), puis */
/* que l'on observe les resulats avec probe prog2 on obtient */

/*      Number of nodes: 27937 */
/*      Number of arrows: 37125 */
/*      Number of terminal nodes: 0 */

/* ce qui valide bien l'absence d'interblocage (pas de noeud terminal) */
/* **** */
```