UNIVERSITE D'ORLEANS

M1 Informatique

Modélisation et vérification de systèmes concurrents

Projet de modélisation et de vérification

Réalisé par:
Bastien FOURNIER
Mamadou DIALLO

Algorithmes	Formules	Résultats	Contre-exemples
1	**CTL**		
	systeme += surete <- AG(J.trm -> (M0.p && M1.p && M2.p));;	Vérifiée	
	systeme += equiteFaible <- AG((M0.p && M1.p && M2.p) -> EF(J.trm));;	Vérifiée	
	systeme += equiteForte <- AG((M0.p && M1.p && M2.p) -> AF(J.trm));;	Vérifiée	
	LTL		
	systeme += surete <- G(J.trm -> (M0.p && M1.p && M2.p));;	Vérifiée	
	systeme += equite <- G((M0.p && M1.p && M2.p) -> F(J.trm));;	Vérifiée	
2	**CTL**		
	systeme += surete <- AG(J.trm -> (M0.p && M1.p && M2.p));;	Non vérifiée	bug = systeme -> J.trm && (M1.a || M2.a);;
	systeme += equiteFaible <- AG((M0.p && M1.p && M2.p) -> EF(J.trm));;	Vérifiée	
	systeme += equiteForte <- AG((M0.p && M1.p && M2.p) -> AF(J.trm));;	Vérifiée	
	LTL		
	systeme += surete <- G(J.trm -> (M0.p && M1.p && M2.p));;	Non vérifiée	
	systeme += equite <- G((M0.p && M1.p && M2.p) -> F(J.trm));;	Vérifiée	

3	**CTL** systeme += surete <- AG(J.trm && coulJ.b -> (M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b));; systeme += equiteFaible <- AG((M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b) -> EF(J.trm));; systeme += equiteForte <- AG((M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b) -> AF(J.trm));; **LTL** systeme += surete <- G(J.trm && coulJ.b -> (M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b));; systeme += equite <- G((M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b) -> F(J.trm));;	Vérifiée Vérifiée Vérifiée Vérifiée Vérifiée	
4	**CTL** systeme += surete <- AG(J.trm && coulJ.b -> (M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b));; systeme += equiteFaible <- AG((M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b) -> EF(J.trm));; systeme += equiteForte <- AG((M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b) -> AF(J.trm)); **LTL** systeme += surete <- G(J.trm && coulJ.b -> (M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b));; systeme += equiteFaible <- G((M0.p && M1.p && M2.p) && (coulM0.b && coulM1.b && coulM2.b) -> F(J.trm));;	Vérifiée Vérifiée Vérifiée Vérifiée Vérifiée	

Contre-exemples:
bug = systeme -> J.trm && (M1.a || M2.a);;
Lors que le jeton se trouve sur la machine m0 et qu'elle(m0) réveille m2 ou m1 ,alors m2 ou m1 reste(nt) active(s) à la fin de l'algorithme.
Toute machine réveillée après avoir passé le jeton reste active à la fin de l'algorithme 2!
Voir algo2_bug_surete.dot

Modèle de l'algorithme 1:

Machine = [a, p]{
    etat=2;
    init=1;
    1 = a;//active
    0 = p;//passive

    0->1[pAa];//passive à active
    1->0[aAp];//active à passive
    0->0[pAp];//passive à passive
    1->1[aAa];//active à active
};;

Jeton= [m0,m1,m2,trm]{

```
    etat = 4;
    init = 2;
    0=m0;//la machine n°0
    1=m1;//la machine n°1
    2=m2;//la machine n°2
    3=trm;//la ternimaison (état terminale)

    2->1 [m2Am1];//M2 envoie le jeton à M1
    1->0 [m1Am0];//M1 envoie le jeton à M0
    0->3 [m0Atrm];//M0 envoie le jeton à l'etat terminale
};;

systeme =<Machine M0, Machine M1, Machine M2, Jeton J> {
        //Envoi de jeton
        //la machine qui envoie le jeton doit être passive d'où le pAp
        <_,_,pAp,m2Am1>;//m2 envoi le jeton à m1 donc elle doit être passive
        <_,pAp,_,m1Am0>;//m1 envoi le jeton à m0 donc elle doit être passive
        <pAp,_,_,m0Atrm>;//m0 envoi le jeton à trm donc elle doit être passive

        <_,_,aAp,_>;//m2 devient passive
        <_,aAp,_,_>;//m1 devient passive
        <aAp,_,_,_>;//m0 devient passive
};;
```

Automates de l'algoritme 1:

```
auto = automaton systeme;;

nonsure= {
    etat=2;
    acc=1;
    init=0;
    0- !J.trm && (M0.p && M1.p && M2.p) -> 0;
    0- (j.trm && !M1.p) && (J.trm && !M2.p) && (J.trm && !M0.p) ->1;
    1- true ->1 [0];
};;

nonequitable= {
    etat=2;
    acc=1;
    init=0;
    0- true -> 0;
    0- !J.trm && M0.p && M1.p && M2.p ->1;
    1- !J.trm ->1 [0];
};;


inter_surete = nonsure && auto;;
inter_eqauite = nonequitable && auto;;

inter_surete = reduce inter_surete;;
inter_eqauite = reduce inter_eqauite;;
```

Modèle de l’algorithme 2 :

Machine = [a, p]{
	etat=2;
	init=1;
	1 = a;//active
	0 = p;//passive

	0->1[pAa];//pAa:passive à active
	1->0[aAp];//active à passive
	0->0[pAp];//passive à passive
	1->1[aAa];//active à active
};;

Jeton= [m0,m1,m2,trm]{
   etat = 4;
   init = 2;
   0=m0;//la machine n°0
   1=m1;//la machine n°1
   2=m2;//la machine n°2
   3=trm;//la ternimaison (état terminale)

   2->2 [Ntrm];//jeton est en m2
   1->1 [Ntrm];//jeton est en m1
   0->0 [Ntrm];//jeton est en m0

   2->1 [m2Am1];//m2 envoie le jeton à m1
   1->0 [m1Am0];//m1 envoie le jeton à m0
   0->3 [m0Atrm];//m0 envoie le jeton à l’état terminale(trm)
};;

systeme =<Machine M0, Machine M1, Machine M2, Jeton J> {

	/**Envoi du jeton de mi+1 à mi:
	 * la machine qui envoie le jeton doit être passive d'où le pAp
	 **/
	<_,_,pAp,m2Am1>;//m2 envoi le jeton à m1
	<_,pAp,_,m1Am0>;//m1 envoi le jeton à m0
	<pAp,_,_,m0Atrm>;//m0 envoi le jeton à trm

	/** Une machine peut passer spontanément de l'état active à l'état passive
	 * le jeton ne doit pas se trouver sur l'état terminale(trm) d'où le Ntrm
	 **/
	<_,_,aAp,Ntrm>;//m2 devient passive
	<_,aAp,_,Ntrm>;//m1 devient passive
	<aAp,_,_,Ntrm>;//m0 devient passive

	/**Envoi de message par les machines actives:
	 * pour cela la machine qui envoie le message doit être active d'où le aAa et
	 * le jeton ne doit pas se trouver sur l'état terminale(trm) d'où le Ntrm

```
        * et la celle qui reçoit le message reste active ou le devient d'où le pAa
        **/

        //Envoi de message par m2 vers m0 ou(exclusif) m1,
        <pAa,_,aAa,Ntrm>;
        <_,pAa,aAa,Ntrm>;

        //Envoi de message par m1 vers m0 ou(exclusif) m2
        <pAa,aAa,_,Ntrm>;
        <_,aAa,pAa,Ntrm>;

        //Envoi de message par m0 vers m1 ou(exclusif) m2
        <aAa,pAa,_,Ntrm>;
        <aAa,_,pAa,Ntrm>;
};;
```

Automates de l'algoritme 2:

```
auto = automaton systeme;;

nonsure= {
    etat=2;
    acc=1;
    init=0;
    0- !J.trm || (M0.p && M1.p && M2.p) -> 0;
    0- (j.trm && !M1.p) || (J.trm && !M2.p) || (J.trm && !M0.p) ->1;
    1- true ->1 [0];
};;

nonequitable= {
    etat=2;
    acc=1;
    init=0;
    0- true -> 0;
    0- !J.trm && M0.p && M1.p && M2.p ->1;
    1- !J.trm ->1 [0];
};;

inter_surete = nonsure && auto;;
inter_eqauite = nonequitable && auto;;

inter_surete = reduce inter_surete;;
inter_eqauite = reduce inter_eqauite;;
```

Modèle de l'algorithme 3:

```
Machine = [a, p]{
        etat=2;
        init=1;
        1 = a;//active
        0 = p;//passive
```

```
        0->1[pAa];//pAa:passive à active
        1->0[aAp];//active à passive
        0->0[pAp];//passive à passive
        1->1[aAa];//aAa:active à active
};;


Jeton= [m0,m1,m2,trm]{
   etat = 4;
   init = 2;

   0=m0;//la machine n°0
   1=m1;//la machine n°1
   2=m2;//la machine n°2
   3=trm;//la ternimaison (état terminale)

   2->2 [Ntrm];//jeton est en m2
   1->1 [Ntrm];//jeton est en m1
   0->0 [Ntrm];//jeton est en m0

   2->1 [m2Am1];//m2 envoie le jeton à m1
   1->0 [m1Am0];//m1 envoie le jeton à m0
   0->3 [m0Atrm];//m0 envoie le jeton à l'etat terminale(trm)
   3->2 [trmAm2];//bouclage
};;

Couleur = [b, n]{
        etat=2;
        init=1;
        1 = b;//la couleur blanche
        0 = n;//la couleur noire

        1->1[bAb, Ab];//bAb: blanche à blanche, Ab: à blanche
        0->1[nAb, Ab];//nAb: noire à blanche
        1->0[bAn, An];//bAn: blanche à noire, An: à noire
        0->0[nAn, An];//nAn: noire à noire
};;

systeme =<Machine M0, Machine M1, Machine M2, Couleur coulM0, Couleur coulM1, Couleur
coulM2, Jeton J, Couleur coulJ> {

        /**Envoi du jeton de mi+1 à mi
         * la machine qui envoie(mi+1) le jeton doit être passive d'où le pAp (Règle 0)
         * si elle est noire elle envoie un jeton noir, sinon elle ne change pas la couleur du jeton Et
                elle devient blanche(Règle 1')
         **/
        //m2 envoi le jeton à m1
        <_,_,pAp,_,_,bAb,m2Am1,_>;//m2 est blanche alors on ne change pas la couleur du jeton
        <_,_,pAp,_,_,nAb,m2Am1,An>;//m2 est noire alors la couleur du jeton reste/devient noire

        //m1 envoi le jeton à m0
        <_,pAp,_,_,bAb,_,m1Am0,_>;//m1 est blanche alors on ne change pas la couleur du jeton
```

```
<_,pAp,_,_,nAb,_,m1Am0,An>;//m1 est noire alors la couleur du jeton reste/devient noire

//m0 envoi le jeton à trm
<pAp,_,_,bAb,_,_,m0Atrm,_>;//m0 est blanche alors on ne change pas la couleur du jeton
<pAp,_,_,nAb,_,_,m0Atrm,An>;//m0 est noire alors la couleur du jeton reste/devient noire



/** Une machine peut passer spontanément de l'état active à l'état passive
 * le jeton ne doit pas se trouver sur l'état terminale(trm) d'où le Ntrm
 **/
<_,_,aAp,_,_,_,Ntrm,_>;//m2 devient passive
<_,aAp,_,_,_,_,Ntrm,_>;//m1 devient passive
<aAp,_,_,_,_,_,Ntrm,_>;//m0 devient passive


/**Les machines actives peuvent envoyer des messages aux autres machines:
 * la machine qui envoie un message doit être active d'où le aAa
 * elle reste/devient noire après avoir envoyé un message (Règle 1') d'où le bAn
 * le jeton ne doit pas se trouver sur l'état terminale(trm) d'où le Ntrm
**/
//Envoi de message par m2 vers m0 ou(exclusif) m1
<pAa,_,aAa,_,_,An,Ntrm,_>;
<_,pAa,aAa,_,_,An,Ntrm,_>;

//Envoi de message par m1 vers m0 ou(exclusif) m2
<pAa,aAa,_,_,An,_,Ntrm,_>;
<_,aAa,pAa,_,An,_,Ntrm,_>;

//Envoi de message par m0 vers m1 ou(exclusif) m2
<aAa,pAa,_,An,_,_,Ntrm,_>;
<aAa,_,pAa,An,_,_,Ntrm,_>;



/**Vérification:
 *si le jeton est noire en m0 alors elle lance une vérification en se blanchissant et en
        envoyant un jeton blanc
 **/
<_,_,_,Ab,_,_,trmAm2,nAb>;

};;
```

Automates de l’algoritme 3:

```
auto = automaton systeme;;

nonsure= {
    etat=2;
    acc=1;
    init=0;
    0- !J.trm || !coulJ.b || (M0.p && M1.p && M2.p && coulM0.b && coulM1.b && coulM2.b) ->
0;
```

```
    0- (J.trm && !M1.p && coulJ.b) || (J.trm && !M2.p && coulJ.b) || (J.trm && coulJ.b && !
coulM0.b) ||
        (J.trm && coulJ.b && !coulM1.b) || (J.trm && coulJ.b && !coulM2.b) || (J.trm && !M0.p
&& coulJ.b) ->1;
    1- true ->1 [0];
};;

nonequitable= {
    etat=2;
    acc=1;
    init=0;
    0- true -> 0;
    0- !J.trm && M0.p && M1.p && M2.p && coulM0.b && coulM1.b && coulM2.b ->1;
    1- !J.trm ->1 [0];
};;

inter_surete = nonsure && auto;;
inter_eqauite = nonequitable && auto;;

inter_surete = reduce inter_surete;;
inter_eqauite = reduce inter_eqauite;;
```

Modèle de l’algorithme 4:

Seul le système de synchronisation diffère de celui de l’algorithme 3.

```
systeme =<Machine M0, Machine M1, Machine M2, Couleur coulM0, Couleur coulM1, Couleur
        coulM2, Jeton J, Couleur coulJ> {

        /**Envoi du jeton de mi+1 à mi
         * la machine qui envoie(mi+1) le jeton doit être passive d'où le pAp (Règle 0)
         * si elle est noire elle envoie un jeton noir, sinon elle ne change pas la couleur du jeton Et
                elle devient blanche(Règle 1')
         **/
        //m2 envoi le jeton à m1
        <_,_,pAp,_,_,bAb,m2Am1,_>;//m2 est blanche alors on ne change pas la couleur du jeton
        <_,_,pAp,_,_,nAb,m2Am1,An>;//m2 est noire alors la couleur du jeton reste/devient noire

        //m1 envoi le jeton à m0
        <_,pAp,_,_,bAb,_,m1Am0,_>;//m1 est blanche alors on ne change pas la couleur du jeton
        <_,pAp,_,_,nAb,_,m1Am0,An>;//m1 est noire alors la couleur du jeton reste/devient noire

        //m0 envoi le jeton à trm
        <pAp,_,_,bAb,_,_,m0Atrm,_>;//m0 est blanche alors on ne change pas la couleur du jeton
        <pAp,_,_,nAb,_,_,m0Atrm,An>;//m0 est noire alors la couleur du jeton reste/devient noire



        /** Une machine peut passer spontanément de l'état active à l'état passive
         * le jeton ne doit pas se trouver sur l'état terminale(trm) d'où le Ntrm
         **/
```

```
<_,_,aAp,_,_,_,Ntrm,_>;//m2 devient passive
<_,aAp,_,_,_,_,Ntrm,_>;//m1 devient passive
<aAp,_,_,_,_,_,Ntrm,_>;//m0 devient passive


/**Les machines actives peuvent envoyer des messages aux autres machines:
 * la machine qui envoie un message doit être active d'où le aAa
 * elle reste/devient noire si elle envoie un message à un destinataire avec un numéro
        supérieur au sien  (Règle 1)
 * le jeton ne doit pas se trouver sur l'état terminale(trm) d'où le Ntrm
**/
//Envoi de message par m2 vers m0 ou(exclusif) m1, donc m2 ne change pas de couleur
<pAa,_,aAa,_,_,_,Ntrm,_>;
<_,pAa,aAa,_,_,_,Ntrm,_>;

//Envoi de message par m1 vers m0 ou(exclusif) m², donc m1 devient noire seulement
        lorsqu'elle envoie à m2
<pAa,aAa,_,_,_,_,Ntrm,_>;
<_,aAa,pAa,_,An,_,Ntrm,_>;

//Envoi de message par m0 vers m1 ou(exclusif) m2 donc m0 devient noire à chaque envoie
        de message
<aAa,pAa,_,An,_,_,Ntrm,_>;
<aAa,_,pAa,An,_,_,Ntrm,_>;



/**Vérification:
 *si le jeton est noire en m0 alors elle lance une vérification en se blanchissant et en
        envoyant un jeton blanc
 **/
<_,_,_,Ab,_,_,trmAm2,nAb>;

};;
```