AltaRica
Synthèse (assistée) d'un contrôleur du niveau d'une cuve

Julien Hannier and Fabien Kuntz

14 mars 2008

Table des matières

Chapitre 1

Cahier des charges

Le système que l'on souhaite concevoir est composé :
- d'un réservoir contenant **toujours** suffisamment d'eau pour alimenter l'exploitation,
- d'une cuve,
- de deux canalisations amont reliant le réservoir à la cuve, et permettant d'amener l'eau à la cuve,
- d'une canalisation aval permettant de vider l'eau de la cuve,
- chaque canalisation est équipée d'une vanne commandable, afin de réguler l'alimentation et la vidange de la cuve,
- d'un contrôleur.

1.1 Détails techniques

1.1.1 La vanne

Les vannes sont toutes de même type, elles possèdent trois niveaux de débits correspondant à trois diamètres d'ouverture : 0 correspond à la vanne fermée, 1 au diamètre intermédiaire et 2 à la vanne complètement ouverte. Les vannes sont commandables par les deux instructions `inc` et `dec` qui respectivement augmente et diminue l'ouverture. Malheureusement, la vanne est sujet à défaillance (rouille), auquel cas le système de commande devient inopérant, la vanne est désormais pour toujours avec la même ouverture.

1.1.2 Les canalisations

Elles sont supposées parfaites.

1.1.3 La Cuve

Elles est munie de quatre capteurs situés à quatre hauteurs qui permettent de délimiter cinq zones. La zone 0 est comprise entre le niveau 0 et le niveau du capteur le plus bas ; la zone 1 est comprise entre ce premier capteur et le second, et ainsi de suite.

1.1.4 Le contrôleur

Il commande les vannes avec les objectifs suivants ordonnés par importance :

1. Le niveau de la cuve ne doit jamais atteindre les zones 0 ou 4.
2. Le débit de la vanne aval doit être le plus important possible.

On fera également l'hypothèse que les commandes ne prennent pas de temps, et qu'entre deux pannes et/ou cycle *temporel*, le contrôleur à toujours le temps de donner au moins un ordre. Réciproquement, on fera l'hypothèse que le système à toujours le temps de réagir entre deux commandes.

1.1.5 Les débits

Les règles suivantes résument l'évolution du niveau de l'eau dans la cuve :
- si $(\sum Vamont_i > \sum Vaval_j)$ alors au temps suivant, le niveau aura augmenté d'une unité.
- si $(\sum Vamont_i < \sum Vaval_j)$ alors au temps suivant, le niveau aura baissé d'une unité.
- si $(\sum Vamont_i = Vaval_j = 0)$ alors au temps suivant, le niveau n'aura pas changé.
- si $(\sum Vamont_i = Vaval_j > 0)$ alors au temps suivant, le niveau pourra :
  - avoir augmenté d'une unité,
  - avoir baissé d'une unité,
  - être resté le même.

1.2 Les spécifications

Le système devra vérifier les propriétés suivantes :

Propriété 1 *En l'absence de panne sur les vannes, et après une éventuelle période initiale la plus courte possible, le niveau de la cuve doit rester dans les zones 1, 2 et 3, et le débit de la vanne aval doit rester maximal.*

Propriété 2 *En présence d'au plus une panne, le niveau de la cuve doit rester dans les zones 1, 2 et 3, et le débit de la vanne aval doit rester le plus souvent possible maximal, et ne jamais devenir nul.*

Propriété 3 *En présence d'au plus deux pannes, le niveau de la cuve doit rester dans les zones 1, 2 et 3, même s'il faut pour cela arrêter définitivement la vanne aval. Néanmoins, le système doit rester le plus possible opérationnel.*

Propriété 4 *En présence d'au plus trois pannes, le niveau de la cuve doit rester dans les zones 1, 2 et 3, même s'il faut pour cela arrêter définitivement la vanne aval. Néanmoins, le système doit rester le plus possible opérationnel.*

1.3 L'étude

L'étude doit aboutir à des résultats et/ou des propositions de solutions du type :
- il est possible/impossible de contrôler.
- il est préférable/inutile de prendre des vannes robustes qui ne tombent jamais en panne.
- il est préférable/inutile de mettre des observateurs sur les vannes pour savoir si elles sont en pannes ou non.

Chapitre 2

Méthodologie de la synthèse assistée

2.1 Les problèmes de synthèse

2.1.1 La synthèse de système

Étant donné une formule $\phi \in \Phi$, un outil de synthèse trouve le *plus grand* $m \in M$ tel que $m \models \phi$

2.1.2 La synthèse de contrôleur

Étant donné une formule $\phi \in \Phi$ et un système $s \in M$, un outil de synthèse de contrôleur trouve le *plus grand* $c \in M$ tel que $(c \times s) \models \phi$

2.1.3 La synthèse assistée de contrôleur

1. Étant donné une formule $\phi \in \Phi$ et un système $s \in M$.
2. Proposer un contrôleur $c_p \in M(A, O)$ ou A est l'ensemble des actions contrôlables et 0 l'ensemble des observations possibles.
3. Répéter
   (a) Calculer avec un vérificateur de modèles le *plus grand* $c \in (c_p \times s)$ tel que $c \models \phi$.
   (b) Si $c = \emptyset$ alors modifier l'environnement (A, O).
   (c) Sinon si $c \neq (c_p \times s)$ alors calculer (avec le vérificateur de modèles ou bien un outil dédié) $c_p = \pi[c_0](c)$.

   jusqu'à $c = (c_p \times s)$ (*le contrôleur c_p est un contrôleur du système.*)

2.2 Méthodologie appliquée au contrôle de la cuve

1. Modélisation des composants de base.
2. Modélisation d'un contrôleur ne contraignant pas les composants.
3. Modèlisation du système controlé.
4. Spécification logique des contrôleurs pour les différentes exigences.
5. Synthèse assistée pour calculer le meilleur contrôleur pouvant contrôler le plus grand nombre d'exigences.

Chapitre 3

Les composants

3.1 La validation des composants

3.1.1 Les spécifications

```
with Cuve, Vanne, VanneRobuste do
 quot()                   >  '$NODENAME.dot';
 dead                     := any_s - src(any_t - self_epsilon);
 notCFC                   := any_t - loop(any_t,any_t);
 show(all)                >  '$NODENAME.res';
 test(dead,0)             >  '$NODENAME.prop';
 test(notCFC,0)           >> '$NODENAME.prop';
done
```

3.2 Le nœud Cuve

3.2.1 Le source AltaRica

```
node Cuve
  flow
    f1, f2, f3 : [0,2];
  state
    niveau : [0,4] : public;
  event
    time;
  trans
    (f1+f2=f3) |- time -> ;
    (f1+f2=f3) & (f3>0) |- time -> niveau := niveau+1;
    (f1+f2=f3) & (f3>0) |- time -> niveau := niveau-1;
    (f1+f2>f3) |- time -> niveau := niveau+1;
    (f1+f2>f3) & (niveau=4) |- time -> ;
    (f1+f2<f3) |- time -> niveau := niveau-1;
    (f1+f2<f3) & (niveau=0)  |- time -> ;
  init
    niveau := 2;
edon
```

3.2.2 La sémantique

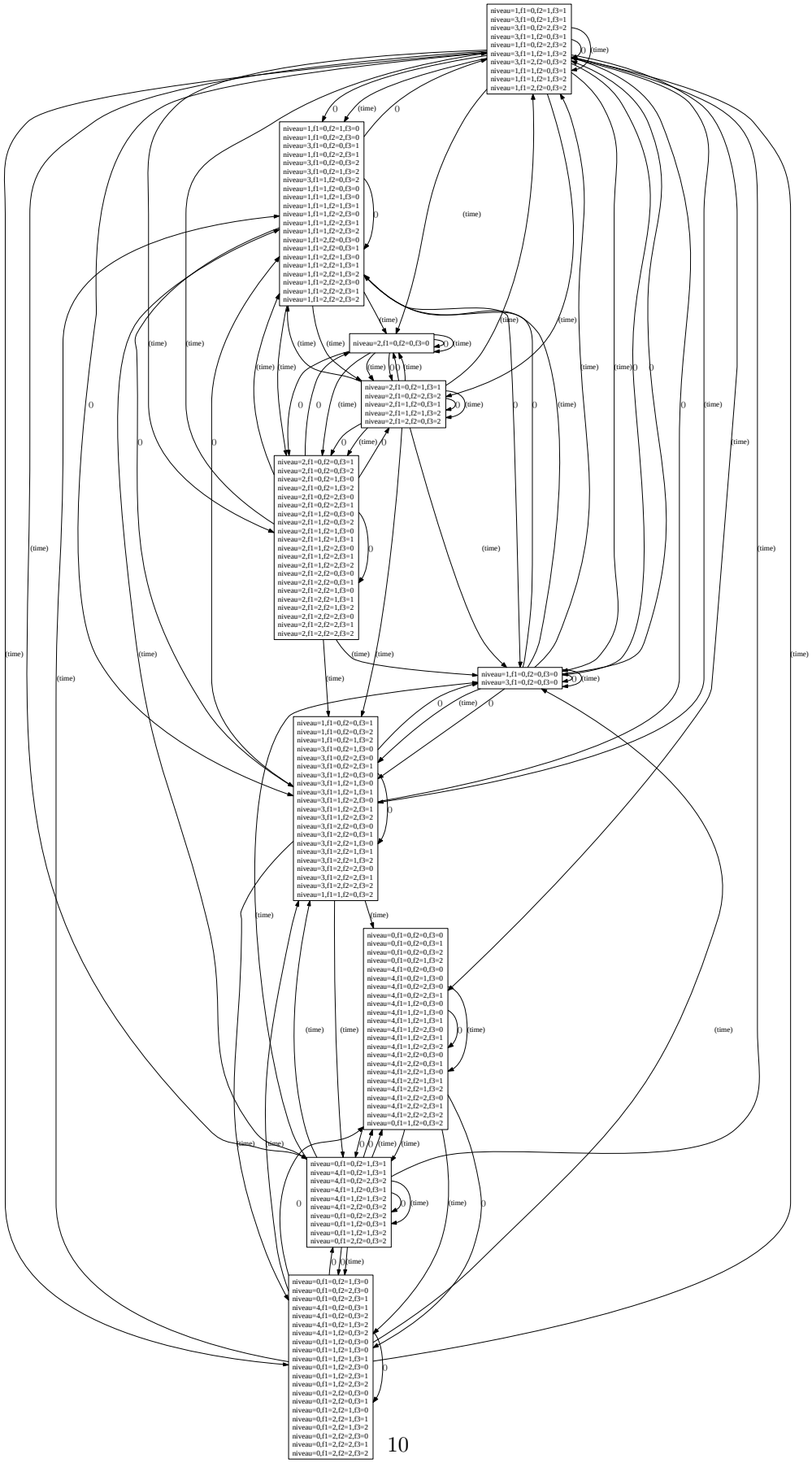

FIG. 3.1 – Le nœud Cuve

3.2.3 Les résultats

```
/*
 * # state properties : 3
 *
 * initial = 27
 * dead = 0
 * any_s = 135
 *
 * # transition properties : 6
 *
 * not_deterministic = 8370
 * epsilon = 3645
 * self = 186
 * any_t = 8370
 * self_epsilon = 135
 * notCFC = 0
 */



TEST(dead=0) [PASSED]
TEST(notCFC=0) [PASSED]
```

3.3 Le nœud Vanne

3.3.1 Le source AltaRica

```
node Vanne
  state
    debit : [0,2] : public;
    on : bool : public;
  flow
    panne : [0,1];
  event
    inc, dec, panne;
  trans
    on    |- inc -> debit := debit+1;
    on    |- dec -> debit := debit-1;
    on    |- panne -> on := false;
    ~on   |- inc, dec -> ;
  assert
    on = (panne=0);
  init
    on := true, debit := 0;
edon
```

3.3.2 La sémantique

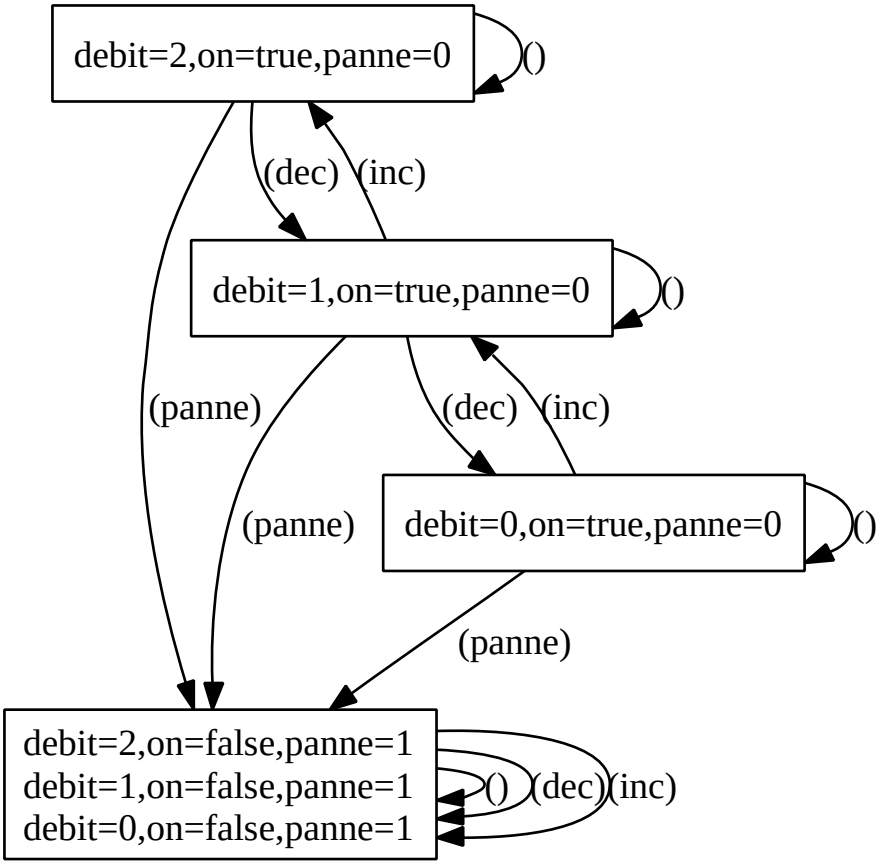

FIG. 3.2 – Le nœud Vanne

3.3.3 Les résultats

```
/*
 * # state properties : 3
 *
 * initial = 1
 * dead = 0
 * any_s = 6
 *
 * # transition properties : 6
 *
 * not_deterministic = 0
 * epsilon = 6
 * self = 12
 * any_t = 19
 * self_epsilon = 6
 * notCFC = 3
 */

TEST(dead=0) [PASSED]
TEST(notCFC=0) [FAILED] actual size = 3
```

Chapitre 4

Le système complet

4.1 Composition d'un système

Un systeme est composé :

de composants représentés qui modèlisent le système *physique* :
- une cuve,
- deux vannes amont,
- une vanne aval.

de composants non représentés dans le modèle car inutiles pour l'analyse réalisée :
- un réservoir,
- les canalisations.

d'un contrôleur représenté dans le modèle par un composant ayant à disposition : d'une part des commandes possibles ; d'autre part une observation partielle du système.

4.2 Le nœud ControleurPermissif

4.2.1 Le source AltaRica

```
node ControleurPermissif
  event
    nop,
    inc1, inc2, inc3, dec1, dec2, dec3,
    inc1inc2, inc1inc3, inc1dec2, inc1dec3,
    inc2inc3, inc2dec1, inc2dec3,
    inc3dec1, inc3dec2,
    dec1dec2, dec1dec3,
    dec2dec3,
    inc1inc2inc3, inc1inc2dec3, inc1dec2inc3, inc1dec2dec3,
    dec1inc2inc3, dec1inc2dec3, dec1dec2inc3, dec1dec2dec3;
  flow
    d1, d2, d3 : [0,2];
    n : [0,4];
  trans
    true  |- nop,
             inc1, inc2, inc3, dec1, dec2, dec3,
             inc1inc2, inc1inc3, inc1dec2, inc1dec3,
             inc2inc3, inc2dec1, inc2dec3,
             inc3dec1, inc3dec2,
             dec1dec2, dec1dec3,
             dec2dec3,
             inc1inc2inc3, inc1inc2dec3, inc1dec2inc3, inc1dec2dec3,
```

```
            dec1inc2inc3, dec1inc2dec3, dec1dec2inc3, dec1dec2dec3
          -> ;
edon
```

4.3 Le nœud System

4.3.1 Le source AltaRica

```
node System
  sub
    V1, V2, V3 : Vanne;
    Cu : Cuve;
    Co : ControleurPermissif; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
```

```
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne>;
    <time, Cu.time, V2.panne>;
    <time, Cu.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

4.4 Le calcul des contrôleurs

L'approche utilisée est la suivante

1. Pour chacune des exigences, calculer le plus grand contrôleur global.
2. Tester si la configuration initiale appartient au contrôleur. Cela indique l'existence ou pas d'un contrôleur.
3. Tester si le contrôleur peut faire des erreurs. Cela indique si ce contrôleur respecte les exigences.
4. Projeter le contrôleur calculé sur le composant *ContrôleurPermissif*.
5. S'il existe, choisir le contrôleur qui satisfait le plus d'exigences, et le substituer au *ContrôleurPermissif*; puis recommencer l'analyse.
6. S'il n'existe pas, recommencer avec un contrôleur qui soit observe plus, soit possède plus d'actions.

4.4.1 Les spécifications des contrôleurs

```
with System, SystemP0, SystemP1, SystemP2, SystemP3, SystemObs, SystemObsP1, SystemObsP2, SystemObsP
 // Des calculs pour valider le modele
 dead                    := any_s - src(any_t - self_epsilon);
 notCFC                  := any_t - loop(any_t,any_t);
 niveau0                 := [Cu.niveau = 0];
 niveau1                 := [Cu.niveau = 1];
 niveau2                 := [Cu.niveau = 2];
 niveau3                 := [Cu.niveau = 3];
 niveau4                 := [Cu.niveau = 4];
 panne                   := label V1.panne | label V2.panne | label V3.panne;
 panne1                  := panne & rtgt([nbPannes=1]);
 panne2                  := panne & rtgt([nbPannes=2]);
 panne3                  := panne & rtgt([nbPannes=3]);
 // les actions controlables
 commande                := label commande;
 controle                := commande;
 // les actions incontrolables
 time                    := label time - (panne1 | panne2 | panne3);
 P0_nonControle          := time;
 P1_nonControle          := panne1 | time;
 P2_nonControle          := panne2 | panne1 | time;
 P3_nonControle          := panne3 | panne2 | panne1 | time;
done
```

```
with System, SystemP0, SystemP1, SystemP2, SystemP3, SystemObs, SystemObsP1, SystemObsP2, SystemObsP
 // les situations redoutees
 ER                      := niveau0 | niveau4 | dead;
done


with SystemRob, SystemRobP0, SystemRobP1, SystemRobP2, SystemRobP3, SystemRetenuOptimise do
 // les situations redoutees incluent les vannes en pannes
 ER                      := niveau0 | niveau4 | dead | [nbPannes>0];
done

/* Les equations classiques
 *  - de stratégies gagnantes (| pour pppf et pour atteindre des etats)
 *  - et de synthèse de contrôleurs (& pour pgpf pour eviter des etats)
 Gagnant           = Gagne |& src(CoupGagnant);
 CoupGagnant       = coup & rtgt(Perdant);
 Perdant           = Perdu |& (src(CoupPerdant) - src(CoupNonPerdant));
 CoupPerdant       = coup & rtgt(Gagnant);
 CoupNonPerdant    = coup - rtgt(Gagnant);
*/
/* Le controleur en phase d'exploitation
 - CtrlER pour eviter ER
 - Le controleur optimal sera calcule ensuite avec les priorites
*/
/* Les quatre proprietes
 - P0 : aucune panne
 - P1 : au plus une panne
 - P2 : au plus deux pannes
 - P3 : au plus trois pannes
*/

// Les controleurs pour la propriete P0
with System, SystemP0, SystemObs, SystemRob, SystemRobP0 do
 // les actions de controle pour eviter ER
 P0_GagneER := [nbPannes=0] - ER;
 P0_PerduER := [nbPannes=0] - ER;
 P0_CtrlER      -=
   controle & rtgt(P0_PerduER &
                   (src(P0_nonControle & rtgt(P0_GagneER & src(P0_CtrlER))) -
                    src(P0_nonControle - rtgt(P0_GagneER & src(P0_CtrlER)))));
 // Le système est-il controlable
 P0_ControlableER            := initial & src(P0_CtrlER);
 // les erreurs de commande possibles
 P0_ErreurControleER         := controle & (rsrc(src(P0_CtrlER)) - P0_CtrlER);
 // Génération des controleurs
 project(any_s, P0_CtrlER, '$NODENAME_Controleur_P0', true, Co)
> 'Alt/$NODENAME_Controleur_P0.alt';
 // sortie des resultats
 show(all)                       >  '$NODENAME_P0.res';
 test(ER,0)                      >  '$NODENAME_P0.prop';
 test(P0_ControlableER,1)           >> '$NODENAME_P0.prop';
 test(P0_ErreurControleER,0)        >> '$NODENAME_P0.prop';
done

with System, SystemP1, SystemObs, SystemObsP1, SystemRob, SystemRobP1 do
```

```
 // La propriete P1
 // les actions de controle pour eviter ER
 P1_GagneER := [nbPannes<2] - ER;
 P1_PerduER := [nbPannes<2] - ER;
 P1_CtrlER      -=
   controle & rtgt(P1_PerduER &
                   (src(P1_nonControle & rtgt(P1_GagneER & src(P1_CtrlER))) -
                    src(P1_nonControle - rtgt(P1_GagneER & src(P1_CtrlER)))));
 // Le système est-il controlable
 P1_ControlableER            := initial & src(P1_CtrlER);
 // les erreurs de commande possibles
 P1_ErreurControleER         := controle & (rsrc(src(P1_CtrlER)) - P1_CtrlER);
 // Génération des controleurs
 project(any_s, P1_CtrlER, '$NODENAME_Controleur_P1', true, Co)
> 'Alt/$NODENAME_Controleur_P1.alt';
 // sortie des resultats
 show(all)                       >  '$NODENAME_P1.res';
 test(ER,0)                      >  '$NODENAME_P1.prop';
 test(P1_ControlableER,1)           >> '$NODENAME_P1.prop';
 test(P1_ErreurControleER,0)        >> '$NODENAME_P1.prop';
done

with System, SystemP2, SystemObs, SystemObsP2, SystemRob, SystemRobP2 do
 // La propriete P2
 // les actions de controle pour eviter ER
 P2_GagneER := [nbPannes<3] - ER;
 P2_PerduER := [nbPannes<3] - ER;
 P2_CtrlER      -=
   controle & rtgt(P2_PerduER &
                   (src(P2_nonControle & rtgt(P2_GagneER & src(P2_CtrlER))) -
                    src(P2_nonControle - rtgt(P2_GagneER & src(P2_CtrlER)))));
 // Le système est-il controlable
 P2_ControlableER            := initial & src(P2_CtrlER);
 // les erreurs de commande possibles
 P2_ErreurControleER         := controle & (rsrc(src(P2_CtrlER)) - P2_CtrlER);
 // Génération des controleurs
 project(any_s, P2_CtrlER, '$NODENAME_Controleur_P2', true, Co)
> 'Alt/$NODENAME_Controleur_P2.alt';
 // sortie des resultats
 show(all)                       >  '$NODENAME_P2.res';
 test(ER,0)                      >  '$NODENAME_P2.prop';
 test(P2_ControlableER,1)           >> '$NODENAME_P2.prop';
 test(P2_ErreurControleER,0)        >> '$NODENAME_P2.prop';
done

with System, SystemP3, SystemObs, SystemObsP3, SystemRob, SystemRobP3, SystemRetenuOptimise do
 // La propriete P3
 // les actions de controle pour eviter ER
 P3_GagneER := [nbPannes<4] - ER;
 P3_PerduER := [nbPannes<4] - ER;
 P3_CtrlER      -=
   controle & rtgt(P3_PerduER &
                   (src(P3_nonControle & rtgt(P3_GagneER & src(P3_CtrlER))) -
                    src(P3_nonControle - rtgt(P3_GagneER & src(P3_CtrlER)))));
 // Le système est-il controlable
 P3_ControlableER            := initial & src(P3_CtrlER);
```

```
 // les erreurs de commande possibles
 P3_ErreurControleER         := controle & (rsrc(src(P3_CtrlER)) - P3_CtrlER);
 // Génération des controleurs
 project(any_s, P3_CtrlER, '$NODENAME_Controleur_P3', true, Co)
> 'Alt/$NODENAME_Controleur_P3.alt';
 // sortie des resultats
 show(all)                       >  '$NODENAME_P3.res';
 test(ER,0)                      >  '$NODENAME_P3.prop';
 test(P3_ControlableER,1)           >> '$NODENAME_P3.prop';
 test(P3_ErreurControleER,0)        >> '$NODENAME_P3.prop';
done
```

4.4.2 Les résultats

Le système sans panne

```
TEST(ER=0) [FAILED] actual size = 617
TEST(P0_ControlableER=1) [PASSED]
TEST(P0_ErreurControleER=0) [FAILED] actual size = 405
```

Le système avec au maximum une panne

```
TEST(ER=0) [FAILED] actual size = 617
TEST(P1_ControlableER=1) [PASSED]
TEST(P1_ErreurControleER=0) [FAILED] actual size = 1931
```

Le système avec au maximum deux pannes

```
TEST(ER=0) [FAILED] actual size = 617
TEST(P2_ControlableER=1) [PASSED]
TEST(P2_ErreurControleER=0) [FAILED] actual size = 1731
```

Le système avec au maximum trois pannes

```
TEST(ER=0) [FAILED] actual size = 617
TEST(P3_ControlableER=1) [PASSED]
TEST(P3_ErreurControleER=0) [FAILED] actual size = 1161
```

4.4.3 Interprétations

Nous venons de modéliser un premier système avec un contrôleur permissif et des vannes basiques. Nous testons ce modèle sur chaque propriété, à savoir avec zéro, une, deux ou trois pannes au maximum.

Le système est contrôlable pour chacune des propriétés mais il contient des erreurs de contrôle qui font qu'il ne peut pas être entièrement contrôlé. De plus, il y a des situations redoutées accessibles pour chaque cas.

Dans le but de mieux contrôler le système, nous allons utiliser les contrôleurs générés et effectuer à nouveau les tests précédents.

4.5 La substitution des contrôleurs calculés

4.5.1 Le nœud SystemP0

```
node SystemP0
  sub
    V1, V2, V3 : Vanne;
```

```
  Cu : Cuve;
  Co : System_Controleur_P0; // ou bien un controleur calcule
state
  controle : bool;
flow
  nbPannes : [0,3];
event
  commande, time;
trans
  controle  |- commande   -> controle := false;
  ~controle |- time       -> controle := true;
assert
  // les liens entre les vannes et les debits de la cuve
  Cu.f1 = V1.debit;
  Cu.f2 = V2.debit;
  Cu.f3 = V3.debit;
  // les observations du controleur
  Co.d1 = V1.debit;
  Co.d2 = V2.debit;
  Co.d3 = V3.debit;
  Co.n  = Cu.niveau;
sync
  // les commandes du controleur sur les debits des vannes
  <commande, Co.inc1, V1.inc>;
  <commande, Co.inc2, V2.inc>;
  <commande, Co.inc3, V3.inc>;
  <commande, Co.dec1, V1.dec>;
  <commande, Co.dec2, V2.dec>;
  <commande, Co.dec3, V3.dec>;
  <commande, Co.inc1inc2, V1.inc, V2.inc>;
  <commande, Co.inc1inc3, V1.inc, V3.inc>;
  <commande, Co.inc1dec2, V1.inc, V2.dec>;
  <commande, Co.inc1dec3, V1.inc, V3.dec>;
  <commande, Co.inc2inc3, V2.inc, V3.inc>;
  <commande, Co.inc2dec1, V2.inc, V1.dec>;
  <commande, Co.inc2dec3, V2.inc, V3.dec>;
  <commande, Co.inc3dec1, V3.inc, V1.dec>;
  <commande, Co.inc3dec2, V3.inc, V2.dec>;
  <commande, Co.dec1dec2, V1.dec, V2.dec>;
  <commande, Co.dec1dec3, V1.dec, V3.dec>;
  <commande, Co.dec2dec3, V2.dec, V3.dec>;
  <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
  <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
  <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
  <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
  <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
  <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
  <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
  <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
  <commande, Co.nop>;
  // les pannes du systeme ont lieu pendant un cycle
  <time, Cu.time, V1.panne>;
  <time, Cu.time, V2.panne>;
  <time, Cu.time, V3.panne>;
  // le temps qui passe
  <time, Cu.time>;
```

```
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

4.5.2 Le nœud SystemP1

```
node SystemP1
  sub
    V1, V2, V3 : Vanne;
    Cu : Cuve;
    Co : System_Controleur_P1; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
```

```
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne>;
    <time, Cu.time, V2.panne>;
    <time, Cu.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

4.5.3 Le nœud SystemP2

```
node SystemP2
  sub
    V1, V2, V3 : Vanne;
    Cu : Cuve;
    Co : System_Controleur_P2; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
```

```
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne>;
    <time, Cu.time, V2.panne>;
    <time, Cu.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

4.5.4 Le nœud SystemP3

```
node SystemP3
  sub
    V1, V2, V3 : Vanne;
    Cu : Cuve;
    Co : System_Controleur_P3; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
  sync
    // les commandes du controleur sur les debits des vannes
```

```
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne>;
    <time, Cu.time, V2.panne>;
    <time, Cu.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

4.5.5 Les résultats

Le système sans panne

```
TEST(ER=0) [FAILED] actual size = 133
TEST(P0_ControlableER=1) [PASSED]
TEST(P0_ErreurControleER=0) [PASSED]
```

Le système avec au maximum une panne

```
TEST(ER=0) [FAILED] actual size = 203
TEST(P1_ControlableER=1) [PASSED]
TEST(P1_ErreurControleER=0) [FAILED] actual size = 483
```

Le système avec au maximum deux pannes

```
TEST(ER=0) [FAILED] actual size = 328
TEST(P2_ControlableER=1) [PASSED]
```

```
TEST(P2_ErreurControleER=0) [FAILED] actual size = 539
```

Le système avec au maximum trois pannes

```
TEST(ER=0) [FAILED] actual size = 72
TEST(P3_ControlableER=1) [PASSED]
TEST(P3_ErreurControleER=0) [FAILED] actual size = 245
```

4.5.6 Interprétations

Grâce à ces tests sur chacune des propriétés, nous remarquons que le système n'est contrôlé que lors de l'absence de pannes. Nous en deduisons donc que se sont les pannes qui rendent le système incontrôlé.

Il nous faut donc créer un nouveau modèle qui gérera les pannes. Ainsi, nous allons voir deux modèles différents. Le premier observera les pannes au niveau des vannes basiques. Le second comportera des vannes qui ne tombent pas en panne.

Chapitre 5

Le système complet avec observation des pannes

5.1 Le nœud ControleurObsPermissif

5.1.1 Le source AltaRica

```
node ControleurObsPermissif
  event
    nop,
    inc1, inc2, inc3, dec1, dec2, dec3,
    inc1inc2, inc1inc3, inc1dec2, inc1dec3,
    inc2inc3, inc2dec1, inc2dec3,
    inc3dec1, inc3dec2,
    dec1dec2, dec1dec3,
    dec2dec3,
    inc1inc2inc3, inc1inc2dec3, inc1dec2inc3, inc1dec2dec3,
    dec1inc2inc3, dec1inc2dec3, dec1dec2inc3, dec1dec2dec3;
  flow
    d1, d2, d3 : [0,2];
    n : [0,4];
    v1, v2, v3 : bool;
  trans
    true  |- nop,
             inc1, inc2, inc3, dec1, dec2, dec3,
             inc1inc2, inc1inc3, inc1dec2, inc1dec3,
             inc2inc3, inc2dec1, inc2dec3,
             inc3dec1, inc3dec2,
             dec1dec2, dec1dec3,
             dec2dec3,
             inc1inc2inc3, inc1inc2dec3, inc1dec2inc3, inc1dec2dec3,
             dec1inc2inc3, dec1inc2dec3, dec1dec2inc3, dec1dec2dec3
          -> ;
edon
```

5.2 Le nœud SystemObs

5.2.1 Le source AltaRica

```
node SystemObs
  sub
```

```
  V1, V2, V3 : Vanne;
  Cu : Cuve;
  Co : ControleurObsPermissif; // ou bien un controleur calcule
state
  controle : bool;
flow
  nbPannes : [0,3];
event
  commande, time;
trans
  controle  |- commande   -> controle := false;
  ~controle |- time       -> controle := true;
assert
  // les liens entre les vannes et les debits de la cuve
  Cu.f1 = V1.debit;
  Cu.f2 = V2.debit;
  Cu.f3 = V3.debit;
  // les observations du controleur
  Co.d1 = V1.debit;
  Co.d2 = V2.debit;
  Co.d3 = V3.debit;
  Co.n  = Cu.niveau;
  // les observations des pannes
  Co.v1 = V1.on;
  Co.v2 = V2.on;
  Co.v3 = V3.on;
sync
  // les commandes du controleur sur les debits des vannes
  <commande, Co.inc1, V1.inc>;
  <commande, Co.inc2, V2.inc>;
  <commande, Co.inc3, V3.inc>;
  <commande, Co.dec1, V1.dec>;
  <commande, Co.dec2, V2.dec>;
  <commande, Co.dec3, V3.dec>;
  <commande, Co.inc1inc2, V1.inc, V2.inc>;
  <commande, Co.inc1inc3, V1.inc, V3.inc>;
  <commande, Co.inc1dec2, V1.inc, V2.dec>;
  <commande, Co.inc1dec3, V1.inc, V3.dec>;
  <commande, Co.inc2inc3, V2.inc, V3.inc>;
  <commande, Co.inc2dec1, V2.inc, V1.dec>;
  <commande, Co.inc2dec3, V2.inc, V3.dec>;
  <commande, Co.inc3dec1, V3.inc, V1.dec>;
  <commande, Co.inc3dec2, V3.inc, V2.dec>;
  <commande, Co.dec1dec2, V1.dec, V2.dec>;
  <commande, Co.dec1dec3, V1.dec, V3.dec>;
  <commande, Co.dec2dec3, V2.dec, V3.dec>;
  <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
  <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
  <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
  <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
  <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
  <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
  <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
  <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
  <commande, Co.nop>;
  // les pannes du systeme ont lieu pendant un cycle
```

```
    <time, Cu.time, V1.panne>;
    <time, Cu.time, V2.panne>;
    <time, Cu.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

5.2.2 Les résultats

Le système sans panne

Il n'est pas traité, car il est dans ce cas inutile d'observer les pannes.

Le système avec au maximum une panne

```
TEST(ER=0) [FAILED] actual size = 617
TEST(P1_ControlableER=1) [PASSED]
TEST(P1_ErreurControleER=0) [FAILED] actual size = 1931
```

Le système avec au maximum deux pannes

```
TEST(ER=0) [FAILED] actual size = 617
TEST(P2_ControlableER=1) [PASSED]
TEST(P2_ErreurControleER=0) [FAILED] actual size = 1731
```

Le système avec au maximum trois pannes

```
TEST(ER=0) [FAILED] actual size = 617
TEST(P3_ControlableER=1) [PASSED]
TEST(P3_ErreurControleER=0) [FAILED] actual size = 1161
```

5.2.3 Interprétations

Cette modélisation du système comporte un contrôleur permissif observant l'état des vannes en permanence, les vannes du premier système n'ayant pas été modifiées. Ici aussi nous testons le modèle sur chacune des propriétés.

Les résultats sont exactement les mêmes que pour le premier modèle. Le système est contrôlable pour chacune des propriétés mais il contient des erreurs de contrôle.

Ainsi, dans le but de mieux contrôler le système, nous allons utiliser les contrôleurs qui viennent d'être générés et effectuer à nouveau les tests précédents.

5.3 La substitution des contrôleurs calculés

5.3.1 Le nœud SystemObsP1

```
node SystemObsP1
  sub
    V1, V2, V3 : Vanne;
    Cu : Cuve;
    Co : SystemObs_Controleur_P1; // ou bien un controleur calcule
  state
```

```
  controle : bool;
flow
  nbPannes : [0,3];
event
  commande, time;
trans
  controle  |- commande   -> controle := false;
  ~controle |- time       -> controle := true;
assert
  // les liens entre les vannes et les debits de la cuve
  Cu.f1 = V1.debit;
  Cu.f2 = V2.debit;
  Cu.f3 = V3.debit;
  // les observations du controleur
  Co.d1 = V1.debit;
  Co.d2 = V2.debit;
  Co.d3 = V3.debit;
  Co.n  = Cu.niveau;
  // les observations des pannes
  Co.v1 = V1.on;
  Co.v2 = V2.on;
  Co.v3 = V3.on;
sync
  // les commandes du controleur sur les debits des vannes
  <commande, Co.inc1, V1.inc>;
  <commande, Co.inc2, V2.inc>;
  <commande, Co.inc3, V3.inc>;
  <commande, Co.dec1, V1.dec>;
  <commande, Co.dec2, V2.dec>;
  <commande, Co.dec3, V3.dec>;
  <commande, Co.inc1inc2, V1.inc, V2.inc>;
  <commande, Co.inc1inc3, V1.inc, V3.inc>;
  <commande, Co.inc1dec2, V1.inc, V2.dec>;
  <commande, Co.inc1dec3, V1.inc, V3.dec>;
  <commande, Co.inc2inc3, V2.inc, V3.inc>;
  <commande, Co.inc2dec1, V2.inc, V1.dec>;
  <commande, Co.inc2dec3, V2.inc, V3.dec>;
  <commande, Co.inc3dec1, V3.inc, V1.dec>;
  <commande, Co.inc3dec2, V3.inc, V2.dec>;
  <commande, Co.dec1dec2, V1.dec, V2.dec>;
  <commande, Co.dec1dec3, V1.dec, V3.dec>;
  <commande, Co.dec2dec3, V2.dec, V3.dec>;
  <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
  <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
  <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
  <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
  <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
  <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
  <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
  <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
  <commande, Co.nop>;
  // les pannes du systeme ont lieu pendant un cycle
  <time, Cu.time, V1.panne>;
  <time, Cu.time, V2.panne>;
  <time, Cu.time, V3.panne>;
  // le temps qui passe
```

```
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

5.3.2 Le nœud SystemObsP2

```
node SystemObsP2
  sub
    V1, V2, V3 : Vanne;
    Cu : Cuve;
    Co : SystemObs_Controleur_P2; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
    // les observations des pannes
    Co.v1 = V1.on;
    Co.v2 = V2.on;
    Co.v3 = V3.on;
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
```

```
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne>;
    <time, Cu.time, V2.panne>;
    <time, Cu.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

5.3.3 Le nœud SystemObsP3

```
node SystemObsP3
  sub
    V1, V2, V3 : Vanne;
    Cu : Cuve;
    Co : SystemObs_Controleur_P3; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
    // les observations des pannes
    Co.v1 = V1.on;
    Co.v2 = V2.on;
    Co.v3 = V3.on;
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
```

```
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne>;
    <time, Cu.time, V2.panne>;
    <time, Cu.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

5.3.4 Les résultats

Le système avec au maximum une panne

```
TEST(ER=0) [FAILED] actual size = 130
TEST(P1_ControlableER=1) [PASSED]
TEST(P1_ErreurControleER=0) [PASSED]
```

Le système avec au maximum deux pannes

```
TEST(ER=0) [FAILED] actual size = 19
TEST(P2_ControlableER=1) [PASSED]
TEST(P2_ErreurControleER=0) [PASSED]
```

Le système avec au maximum trois pannes

```
TEST(ER=0) [PASSED]
TEST(P3_ControlableER=1) [PASSED]
TEST(P3_ErreurControleER=0) [PASSED]
```

5.3.5 Interprétations

Grâce à ces tests sur chacune des propriétés avec pannes, nous remarquons que le système est totalement contrôlé dans tous les cas. De plus, il n'y a plus aucune situation redoutée accessible dans le système avec au maximum trois pannes.

Ce modèle est meilleur que le premier que nous avons testé mais il subsiste des erreurs de controle dans certains cas. Voyons ce qu'il en est pour le modèle avec les vannes robustes.

Chapitre 6

Le système complet avec des vannes robustes

6.1 Le nœud VanneRobuste

6.1.1 Le source AltaRica

```
node VanneRobuste
  state
    debit : [0,2] : public;
    on : bool : public;
    age : [0,2] : public;
  flow
    panne : [0,1];
  event
    inc, dec, panne, time;
  trans
    on           |- inc -> debit := debit+1, age :=0;
    on           |- dec -> debit := debit-1, age :=0;
    on & (age=2) |- panne -> on := false;
    on & (age<2) |- time -> age := age+1;
    on & (age=2) |- time -> ;
    ~on          |- inc, dec, time -> ;
  assert
    on = (panne=0);
  init
    on := true, debit := 0, age := 0;
edon
```

6.1.2 La sémantique

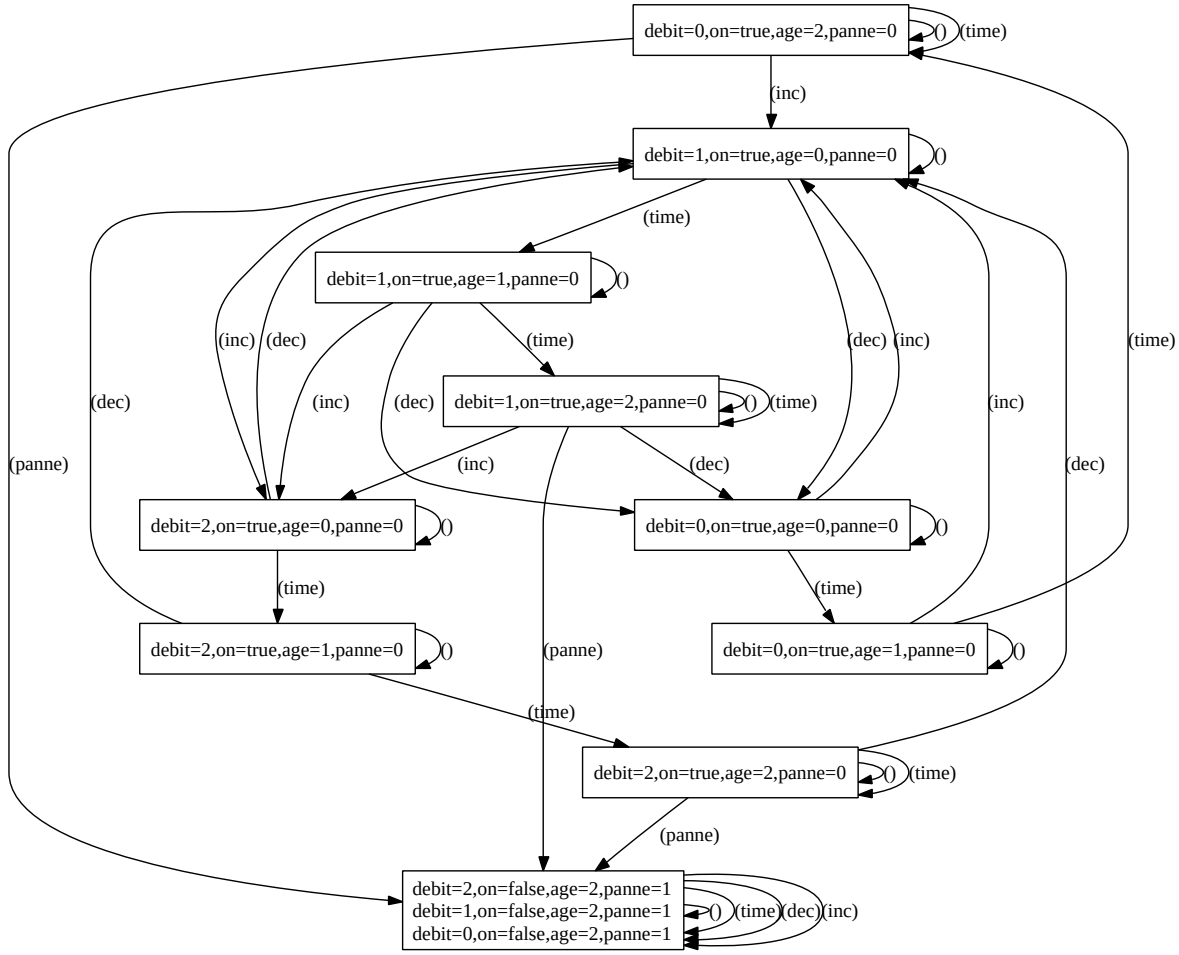

FIG. 6.1 – Le nœud VanneRobuste

6.1.3 Les résultats

```
/*
 * # state properties : 3
 *
 * initial = 1
 * dead = 0
 * any_s = 12
 *
 * # transition properties : 6
 *
 * not_deterministic = 0
 * epsilon = 12
 * self = 24
 * any_t = 45
 * self_epsilon = 12
 * notCFC = 3
 */

TEST(dead=0) [PASSED]
TEST(notCFC=0) [FAILED] actual size = 3
```

6.1.4 Interprétations

Le noeud vanne robuste ne contient aucun deadlock comme la vanne simple. Il en est de même pour les trois composantes fortement connexes que nous retrouvons ici.

Cela nous permet de l'utiliser dans ce modèle car nous savons qu'elle ne posera aucun problème autre que les précédents.

6.2 Le nœud ControleurRobPermissif

6.2.1 Le source AltaRica

```
node ControleurRobPermissif
  event
    nop,
    inc1, inc2, inc3, dec1, dec2, dec3,
    inc1inc2, inc1inc3, inc1dec2, inc1dec3,
    inc2inc3, inc2dec1, inc2dec3,
    inc3dec1, inc3dec2,
    dec1dec2, dec1dec3,
    dec2dec3,
    inc1inc2inc3, inc1inc2dec3, inc1dec2inc3, inc1dec2dec3,
    dec1inc2inc3, dec1inc2dec3, dec1dec2inc3, dec1dec2dec3;
  flow
    d1, d2, d3 : [0,2];
    n : [0,4];
    a1, a2, a3 : [0,2];
  trans
    true  |- nop,
             inc1, inc2, inc3, dec1, dec2, dec3,
             inc1inc2, inc1inc3, inc1dec2, inc1dec3,
             inc2inc3, inc2dec1, inc2dec3,
             inc3dec1, inc3dec2,
             dec1dec2, dec1dec3,
             dec2dec3,
             inc1inc2inc3, inc1inc2dec3, inc1dec2inc3, inc1dec2dec3,
             dec1inc2inc3, dec1inc2dec3, dec1dec2inc3, dec1dec2dec3
          -> ;
edon
```

6.3 Le nœud SystemRob

6.3.1 Le source AltaRica

```
node SystemRob
  sub
    V1, V2, V3 : VanneRobuste;
    Cu : Cuve;
    Co : ControleurRobPermissif; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
```

```
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
    // les observations des ages des vannes
    Co.a1 = V1.age;
    Co.a2 = V2.age;
    Co.a3 = V3.age;
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne, V2.time, V3.time>;
    <time, Cu.time, V1.time, V2.panne, V3.time>;
    <time, Cu.time, V1.time, V2.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time, V1.time, V2.time, V3.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

6.3.2 Les résultats

Le système sans panne

```
TEST(ER=0) [FAILED] actual size = 6650
TEST(P0_ControlableER=1) [PASSED]
TEST(P0_ErreurControleER=0) [FAILED] actual size = 3014
```

Le système avec au maximum une panne

```
TEST(ER=0) [FAILED] actual size = 6650
TEST(P1_ControlableER=1) [PASSED]
TEST(P1_ErreurControleER=0) [FAILED] actual size = 4385
```

Le système avec au maximum deux pannes

```
TEST(ER=0) [FAILED] actual size = 6650
TEST(P2_ControlableER=1) [PASSED]
TEST(P2_ErreurControleER=0) [FAILED] actual size = 4385
```

Le système avec au maximum trois pannes

```
TEST(ER=0) [FAILED] actual size = 6650
TEST(P3_ControlableER=1) [PASSED]
TEST(P3_ErreurControleER=0) [FAILED] actual size = 4385
```

6.3.3 Interprétations

Cette modélisation du système comporte donc un contrôleur permissif et des vannes ne tombant pratiquement jamais en panne. Pour ce faire nous utilisons des vannes robustes présentées plus haut. Ici encore nous testons ce modèle sur chacune des propriétés.

Les résultats sont exactement les mêmes que pour le premier et le second modèle. Le système est contrôlable pour chacune des propriétés mais il contient des erreurs de contrôle. De plus, il y a toujours des situations redoutées accessibles dans chaque cas.

Ainsi, dans le but de mieux contrôler le système, nous allons utiliser les contrôleurs qui viennent d'être générés et effectuer à nouveau les tests précédents.

6.4 La substitution des contrôleurs calculés

6.4.1 Le nœud SystemRobP3

```
node SystemRobP3
  sub
    V1, V2, V3 : VanneRobuste;
    Cu : Cuve;
    Co : SystemRob_Controleur_P3; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
```

```
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
    // les observations des ages des vannes
    Co.a1 = V1.age;
    Co.a2 = V2.age;
    Co.a3 = V3.age;
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne, V2.time, V3.time>;
    <time, Cu.time, V1.time, V2.panne, V3.time>;
    <time, Cu.time, V1.time, V2.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time, V1.time, V2.time, V3.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

6.4.2 Les résultats

Le système sans panne

```
TEST(ER=0) [FAILED] actual size = 1962
TEST(P0_ControlableER=1) [PASSED]
TEST(P0_ErreurControleER=0) [PASSED]
```

Le système avec au maximum une panne

```
TEST(ER=0) [PASSED]
TEST(P1_ControlableER=1) [PASSED]
TEST(P1_ErreurControleER=0) [PASSED]
```

Le système avec au maximum deux pannes

```
TEST(ER=0) [PASSED]
TEST(P2_ControlableER=1) [PASSED]
TEST(P2_ErreurControleER=0) [PASSED]
```

Le système avec au maximum trois pannes

```
TEST(ER=0) [PASSED]
TEST(P3_ControlableER=1) [PASSED]
TEST(P3_ErreurControleER=0) [PASSED]
```

6.4.3 Interprétations

Grâce à ces tests sur chacune des propriétés avec et sans pannes, nous remarquons que le système est totalement contrôlé dans tous les cas. De plus, il n'y a plus aucune situation redoutée accessible dans les propriétés avec pannes. Il en reste seulement dans le cas sans pannes.

Ce modèle est bien meilleur que le premier ainsi que le second que nous avons testé. Cependant, il reste des situations redoutées accessibles quand il n'y a pas de pannes.

Chapitre 7

Préconisations pour le contrôle d'une cuve

7.1 Resultats des tests des differents modeles

Nous venons de tester trois modèles différents par leurs caractéristiques.

Le premier, très simple, ne peut pas être contrôlé dans tous les cas et il contient de nombreuses situations redoutées.

Le second peut être contrôlé dans tous les cas mais il a encore des situations redoutées pour les systemes avec une et deux pannes au maximum.

Le troisième, quant à lui, peut aussi être contrôlé dans tous les cas et ne contient des situations redoutées que pour le cas où il n'y a pas de panne.

7.2 Conclusion des analyses des resultats

Il semble donc assez clair que le contrôleur le plus intéressant est le contrôleur résultat de la projection du système robuste P3 sur son noeud Co.

C'est donc ce dernier que nous avons choisi de reutiliser et d'optimiser.

Chapitre 8

L'optimisation du contrôleur retenu

8.1 Le nœud ControleurRetenuOptimise

```
/*
 * This node is the result of the projection of the node 'SystemRob'
 * on its subnode 'Co'.
 */
node ControleurRetenuOptimise
 flow
  d1 : [0,2];
  d2 : [0,2];
  d3 : [0,2];
  n : [0,4];
  a1 : [0,2];
  a2 : [0,2];
  a3 : [0,2];
 event
   nop; inc1; inc2; inc3; dec1; dec2; dec3; inc1inc2; inc1inc3; inc1dec2; inc1dec3; inc2inc3; inc2de
   // Priorities to optimatise the output through V3
   inc3 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc1inc3 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc2inc3 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc3dec1 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc3dec2 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc1inc2inc3 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc1dec2inc3 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   dec1inc2inc3 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   dec1dec2inc3 > {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc1dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc2dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   dec1dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   dec2dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc1inc2dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   inc1dec2dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   dec1inc2dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
   dec1dec2dec3 < {nop, inc1, inc2, dec1, dec2, inc1inc2, inc1dec2, inc2dec1, dec1dec2};
 trans
   n = 1 and (d3 = 0 and (d2 = 0 and (d1 = 0 or d1 = 1) and a3 = 1 and a2 = 1 and a1 = 1 or d2 = 1 a
   n = 1 and (d3 = 0 and (d2 = 0 and (d1 = 0 and a3 = 1 and a2 = 1 and (a1 = 1 or a1 = 2) or d1 = 1
   n = 1 and (d3 = 0 and (d2 = 0 and ((d1 = 0 or d1 = 1) and a3 = 1 and (a2 = 1 or a2 = 2) and a1 =
   n = 1 and (d3 = 0 and (d2 = 1 and (d1 = 1 and (a3 = 1 or a3 = 2) and a2 = 1 and a1 = 1 or d1 = 2
```

```
   n = 1 and (d3 = 0 and (d2 = 0 and d1 = 1 and a3 = 1 and a2 = 1 and a1 = 1 or d2 = 1 and (d1 = 1 a
   n = 1 and (d3 = 0 and (d2 = 1 and (d1 = 0 and a3 = 1 and a2 = 1 and a1 = 1 or d1 = 1 and a3 = 1 a
   n = 1 and (d3 = 1 and (d2 = 0 and ((d1 = 0 or d1 = 1) and (a3 = 1 or a3 = 2) and a2 = 1 and a1 =
   n = 1 and (d3 = 0 and (d2 = 0 and (d1 = 0 and a3 = 1 and (a2 = 1 or a2 = 2) and (a1 = 1 or a1 = 2
   n = 1 and (d3 = 0 and (d2 = 0 and d1 = 1 and a3 = 1 and a2 = 1 and a1 = 1 or d2 = 1 and (d1 = 0 a
   n = 1 and (d3 = 0 and (d2 = 1 and (d1 = 0 and a3 = 1 and (a2 = 1 and (a1 = 1 or a1 = 2) or a2 = 2
   n = 1 and (d3 = 1 and ((d2 = 0 or d2 = 1) and (d1 = 0 or d1 = 1) and (a3 = 1 or a3 = 2) and a2 =
   n = 1 and (d3 = 0 and (d2 = 0 and (d1 = 1 and (a3 = 1 and (a2 = 1 or a2 = 2) and a1 = 1 or a3 = 2
   n = 1 and (d3 = 0 and (d2 = 0 and (d1 = 1 and a3 = 1 and (a2 = 1 and a1 = 1 or a2 = 2 and (a1 = 1
   n = 1 and (d3 = 1 and (d2 = 0 and ((d1 = 0 or d1 = 1) and (a3 = 1 or a3 = 2) and (a2 = 1 or a2 =
   n = 1 and (d3 = 0 and (d2 = 1 and d1 = 2 and a3 = 1 and a2 = 1 and a1 = 1 or d2 = 2 and d1 = 1 an
   n = 1 and (d3 = 0 and (d2 = 1 and d1 = 2 and a3 = 1 and a2 = 1 and a1 = 1 or d2 = 2 and d1 = 1 an
   n = 1 and (d3 = 0 and (d2 = 1 and (d1 = 1 and a3 = 1 and (a2 = 1 and (a1 = 1 or a1 = 2) or a2 = 2
   n = 1 and (d3 = 1 and (d2 = 0 and (d1 = 1 and (a3 = 1 or a3 = 2) and a2 = 1 and (a1 = 1 or a1 = 2
   n = 1 and (d3 = 1 and (d2 = 1 and ((d1 = 0 or d1 = 1) and (a3 = 1 or a3 = 2) and (a2 = 1 or a2 =
   n = 1 and (d3 = 0 and (d2 = 0 and (d1 = 0 and (a3 = 1 or a3 = 2) and (a2 = 1 or a2 = 2) and (a1 =
   n = 1 and (d3 = 1 and (d2 = 0 and (d1 = 0 or d1 = 1) and (a3 = 1 or a3 = 2) and (a2 = 1 or a2 = 2
   n = 1 and (d3 = 0 and (d2 = 1 and d1 = 1 and (a3 = 1 and (a2 = 1 and (a1 = 1 or a1 = 2) or a2 = 2
   n = 1 and (d3 = 1 and (d2 = 1 and (d1 = 0 and (a3 = 1 or a3 = 2) and (a2 = 1 or a2 = 2) and (a1 =
   n = 1 and (d3 = 0 and (d2 = 0 and d1 = 2 and a3 = 1 and a2 = 2 and a1 = 1 or d2 = 1 and (d1 = 1 a
   n = 1 and (d3 = 1 and (d2 = 0 and (d1 = 1 and (a3 = 1 or a3 = 2) and (a2 = 1 or a2 = 2) and (a1 =
   n = 2 and (d3 = 0 and (d2 = 1 and (d1 = 1 or d1 = 2) and (a3 = 1 or a3 = 2) and (a2 = 1 or a2 = 2
   n = 1 and (d3 = 1 and (d2 = 1 and (d1 = 1 and (a3 = 1 and (a2 = 1 or a2 = 2) and (a1 = 1 or a1 =
edon
```

8.2 Le nœud SystemRetenuOptimise

```
node SystemRetenuOptimise
  sub
    V1, V2, V3 : VanneRobuste;
    Cu : Cuve;
    Co : ControleurRetenuOptimise; // ou bien un controleur calcule
  state
    controle : bool;
  flow
    nbPannes : [0,3];
  event
    commande, time;
  trans
    controle  |- commande   -> controle := false;
    ~controle |- time       -> controle := true;
  assert
    // les liens entre les vannes et les debits de la cuve
    Cu.f1 = V1.debit;
    Cu.f2 = V2.debit;
    Cu.f3 = V3.debit;
    // les observations du controleur
    Co.d1 = V1.debit;
    Co.d2 = V2.debit;
    Co.d3 = V3.debit;
    Co.n  = Cu.niveau;
    // les observations des ages des vannes
    Co.a1 = V1.age;
    Co.a2 = V2.age;
    Co.a3 = V3.age;
```

```
  sync
    // les commandes du controleur sur les debits des vannes
    <commande, Co.inc1, V1.inc>;
    <commande, Co.inc2, V2.inc>;
    <commande, Co.inc3, V3.inc>;
    <commande, Co.dec1, V1.dec>;
    <commande, Co.dec2, V2.dec>;
    <commande, Co.dec3, V3.dec>;
    <commande, Co.inc1inc2, V1.inc, V2.inc>;
    <commande, Co.inc1inc3, V1.inc, V3.inc>;
    <commande, Co.inc1dec2, V1.inc, V2.dec>;
    <commande, Co.inc1dec3, V1.inc, V3.dec>;
    <commande, Co.inc2inc3, V2.inc, V3.inc>;
    <commande, Co.inc2dec1, V2.inc, V1.dec>;
    <commande, Co.inc2dec3, V2.inc, V3.dec>;
    <commande, Co.inc3dec1, V3.inc, V1.dec>;
    <commande, Co.inc3dec2, V3.inc, V2.dec>;
    <commande, Co.dec1dec2, V1.dec, V2.dec>;
    <commande, Co.dec1dec3, V1.dec, V3.dec>;
    <commande, Co.dec2dec3, V2.dec, V3.dec>;
    <commande, Co.inc1inc2inc3, V1.inc, V2.inc, V3.inc>;
    <commande, Co.inc1inc2dec3, V1.inc, V2.inc, V3.dec>;
    <commande, Co.inc1dec2inc3, V1.inc, V2.dec, V3.inc>;
    <commande, Co.inc1dec2dec3, V1.inc, V2.dec, V3.dec>;
    <commande, Co.dec1inc2inc3, V1.dec, V2.inc, V3.inc>;
    <commande, Co.dec1inc2dec3, V1.dec, V2.inc, V3.dec>;
    <commande, Co.dec1dec2inc3, V1.dec, V2.dec, V3.inc>;
    <commande, Co.dec1dec2dec3, V1.dec, V2.dec, V3.dec>;
    <commande, Co.nop>;
    // les pannes du systeme ont lieu pendant un cycle
    <time, Cu.time, V1.panne, V2.time, V3.time>;
    <time, Cu.time, V1.time, V2.panne, V3.time>;
    <time, Cu.time, V1.time, V2.time, V3.panne>;
    // le temps qui passe
    <time, Cu.time, V1.time, V2.time, V3.time>;
  assert
    nbPannes = (V1.panne + V2.panne + V3.panne);
  init
    controle := true;
edon
```

8.3 Les résultats

```
TEST(ER=0) [PASSED]
TEST(P3_ControlableER=1) [PASSED]
TEST(P3_ErreurControleER=0) [PASSED]
```

8.3.1 Interprétations

Nous venons donc de réutiliser le contrôleur que nous avons retenu en l'optimisant pour assurer un débit maximal en aval.

Pour ce faire, nous avons utilisé les priorites AltaRica. Maximiser le débit de la vanne de sortie (vanne 3) équivaut à privilégier l'augmentation du débit de la vanne par rapport aux autres actions, et privilégier la stagnation du débit par rapport à sa diminution (*inc3 > tout ou il n'y a pas inc3 ou dec3

$>$ *dec3).

Il s'avère que ce nouveau modèle répond parfaitement aux attentes du cahier des charges. Nous obtenons un débit maximal tout en contrôlant parfaitement le système et en évitant toutes les situations redoutées.

Mission accomplie.