Nous utilisons le logiciel PRISM qui est un "model checker" des modèles probabilistes. Nous allons étudier des modèles de fiabilité et de performances vus en cours et en TDs. Vous pouvez trouver une large documentation de ce logiciel dans la page [http://www.prismmodelchecker.org](http://www.prismmodelchecker.org/).

1- Créez un répertoire avec votre nom

2- Vous lancez Prism: dans Applications/prism-4.0.3-osx32/bin/xprism

3- Vous spécifiez le premier modèle: Il y a deux composantes dans le système; C1 peut tomber en panne selon une loi exponentielle de taux lambda1, C2 peut tomber en panne selon une loi exponentielle de taux lambda2. Il y a 2 réparateurs et les temps de réparations sont exponentiels de taux: pour C1= mu1, pour C2= mu2.

Voici le modèle avec le langage de spécification PRISM:

ctmc

const double lambda1=0.1;

const double lambda2=0.15;

const double mu1=0.1;

const double mu2=0.11;

module M1

C1:[0..1] init 1; // etat de la premiere composante

[ ] (C1=1) -> lambda1: (C1'= 0); // avec taux lambda1 1-->0

[ ] (C1=0) -> mu1: (C1'= 1); // avec taux mu1 0-->1

endmodule

module M2

C2:[0..1] init 1; // etat de la deuxieme composante

[ ] (C2=1) -> lambda2: (C2'= 0); // avec taux lambda2 1-->0

[ ] (C2=0) -> mu2: (C2'= 1); // avec taux mu2 0-->1

endmodule

Sauvegardez le modèle dans votre répertoire sous le nom **modele1.sm**

4- Dans le menu Model , faites build model. Vous verrez le nombre d'états et le nombre de transitions dans le coin gauche en bas. Visualiser les états et le générateur infinitésimale par le menu Model->View. Pour ce modèle, les états sont

(C1,C2)

0:(0,0) // état 0 indique C1=0, C2=0

1:(0,1)

2:(1,0)

3:(1,1)

la matrice:

*Exporting transition matrix in plain text format below:*

4 8 //nombre d'états nombre de transitions

0 1 0.1 // la transition de l'état 0 à 1 avec un taux 3.

0 1 0.11

0 2 0.1

1 0 0.15

1 3 0.1

2 0 0.1

2 3 0.11

3 1 0.1

3 2 0.15

Donnez le générateur pour ce modèle et comparez avec celui du TD. Pour revenir, vous cliquez sur l'onglet model à gauche en bas.

5-Il est également possible de définir dans un seul module

module M

C1:[0..1] init 1; // etat de la premiere composante

C2:[0..1] init 1; // etat de la deuxieme composante

[] (C1=1) -> lambda1: (C1'= 0); // avec taux lambda1 1-->0

[] (C1=0) -> mu1: (C1'= 1); // avec taux mu1 0-->

[] (C2=1) -> lambda2: (C2'= 0); // avec taux lambda2 1-->0

[] (C2=0) -> mu2: (C2'= 1); // avec taux mu2 0-->1

endmodule

Sauvegardez le modèle dans votre répertoire sous le nom **modele2.sm.** Construisez le générateur de ce modèle et comparez avec celui du **modele1.sm.** Que constatez-vous?

6-Vous ajoutez la commande suivante dans le modèle (en dehors des modules). Nous qualifions "up", si les valeurs de C1+C2 >=1. Il s'agit donc un système parallèle.

label "up"= C1+C2>=1;

label "down"= C1+C2=0;

Nous pouvons calculer la distribution stationnaire: dans le menu Model ->Compute

Quelle est la disponibilité de ce système à l'état stationnaire ?

*Vous pouvez utiliser l'opérateur "steady-state (S)". Ouvrez l'onglet properties, ajouter un opérateur en cliquant sur Add pour entrer* *S=?[("up" )]. Cet opérateur calcule la probabilité des états up lorsque le système se trouve à l'équilibre. Allez dans le menu properties et cliquez sur verify.*

6-Vous multipliez tous les taux de transitions par 10, (par exemple lambda1=1 au lieu de 0.1). Quelle est la disponibilité de ce système à l'état stationnaire?

7-Nous étudions maintenant les probabilités transitoires. Mettez les taux à leurs anciennes valeurs. Calculez les probabilités transitoires des états où il n'y a plus de composant opérationnel aux instants t=1.0, t=1.5, t=2.0,t=2.5, … t=15.

*Ouvrez l'onglet propriétés et en cliquant sur le rectangle*

*-Constants choisissez add pour ajouter T comme constant*

*-Properties choisissez add pour définir la propriété (la probabilité d'être dans un état up à l'instant T)*

*P=?[(true) U [T,T] ("up")]*

*-cliquez sur le rectangle Experiments, choisissez new experiment*

*choisissez range Start 0 End 15 Step 0.5, puis cliquez Okay pour dessiner l'évolution de cette propriété en fonction de T.*

8-Vous multipliez les taux par 10 et refaites les calculs précédents.

Que constatez-vous?

8- Modifiez *label "up"* pour calculer la disponibilité stationnaire d'un système en séries.

9- Nous considérons maintenant un système avec 5 composants identiques (les mêmes taux de panne et réparation) et chaque composant tombe en panne avec taux 0.01. Il y a un seul réparateur et la durée moyenne d'une réparation est 50 unités de temps. Il faut au moins deux composants opérationnels pour que le système fonctionne. Calculez la disponibilité stationnaire de ce système.

10- Nous considérons un système à 2 composantes avec les taux de panne et les taux de réparation différents. Il y a un seul agent pour faire les réparations. Une spécification possible de ce modèle est donnée dans la suite. [etiquette] sont des transitions synchronisées.

ctmc

const double lambda1=1;

const double lambda2=1.5;

const double mu1=2;

const double mu2=2.5;

module M1

C1:[0..1] init 1; // etat de la premiere composante

[finrep1C2UP ] (C1=0) &(agent=1) -> mu1: (C1'= 1); //

[finrep1C2DOWN ] (C1=0) &(agent=1)-> mu1: (C1'= 1); //

[panneC1rep0] (C1=1) -> lambda1: (C1'=0);

[panneC1rep2] (C1=1) -> lambda1: (C1'=0);

endmodule

module M2

C2:[0..1] init 1; // etat de la deuxieme composante

[finrep2C1UP ] (C2=0) &(agent=2) -> mu2: (C2'= 1); //

[finrep2C1DOWN ] (C2=0) &(agent=2)-> mu2: (C2'= 1); //

[panneC2rep0] (C2=1) -> lambda2: (C2'=0);

[panneC2rep1] (C2=1) -> lambda2: (C2'=0);

endmodule

module reparateur

agent:[0..2] init 0;

[panneC1rep0] (agent=0) -> 1: (agent'=1);

[panneC1rep2] (agent=2) -> 1: (agent'=2);

[panneC2rep0] (agent=0) -> 1: (agent'=2);

[panneC2rep1] (agent=1) -> 1: (agent'=1);

[finrep1C2UP ] (C2=1) -> 1: (agent'= 0);

[finrep1C2DOWN ] (C2=0) -> 1: (agent'= 2);

[finrep2C1UP ] (C1=1) -> 1: (agent'= 0);

[finrep2C1DOWN ] (C1=0) -> 1: (agent'= 1);

endmodule

Sauvegardez ce programme sous le nom **module3.sm**. Calculer la disponibilité stationnaire de ce système (1 seule composante est suffisante).

11- Vous reprenez le programme **module2.sm.** Vous modifiez le programme pour que le taux de réparation mu2 soit une distribution Erlang au lieu d'une distribution exponentielle de même moyenne. Comparez la disponibilité stationnaire de ce système avec celle du **module2.sm.**

12- On considère un système informatique (vu en cours). Il y a deux processeurs P1 et P2 et chacun a une unité de mémoire locale LM1 et LM2. Il existe aussi une unité de la mémoire globale GM. Premièrement on suppose qu'il n'y a pas de réparation. En moyenne P1 fonctionne pendant 10 unités de temps, P2 pendant 8 unités de temps, LM1 et LM2 pendant 5 unités de temps, et GM 1à unités de temps. Les distributions sont exponentielles.

Le système est disponible si au moins un processeur fonctionne soit avec son unité de mémoire locale ou avec la mémoire globale.

Calculez la disponibilité stationnaire de ce système.

Calculez les disponibilités transitoires pour T=0 … 50

Ajouter un réparateur dans votre modèle. Les priorités pour la réparation (s'il y a plusieurs réparation possibles): 1-GM, 2-P1, 3-P2, 4-LM1, 5-LM2