Université de Lorraine DIPLOME: Telecom Nancy 2A - Apprentissage Épreuve: ASPD épreuve pratique	Durée du sujet : 1 h 30 Date : Vendredi 4 juin 2021 entre 16 h 00 et 17 h 30 Nom du rédacteur : Dominique Méry Documents personnels autorisés

Epreuve TP ASPD

Vous enverrez votre archive nom-prenom-aspd2021.zip; vous attendrez aant de partir un accusé de réception de la part de Dominique Méry avant de quitter la salle d'épreuve.

Exercice 1

La figure 1 est un réseau de Petri modélisant un système.

Question 1.1 *Traduire le réseau de Petri sous la forme d'un module TLA en utilisant le fichier apppetri2021.tla. En particulier, il faut compléter l'initialisation.*

Question 1.2 *Est-ce que le réseau peut atteindre un point de deadlock? Expliquez votre réponse.*

Question 1.3 *Donner deux propriétés de sûreté vérifiées par ce réseau.*

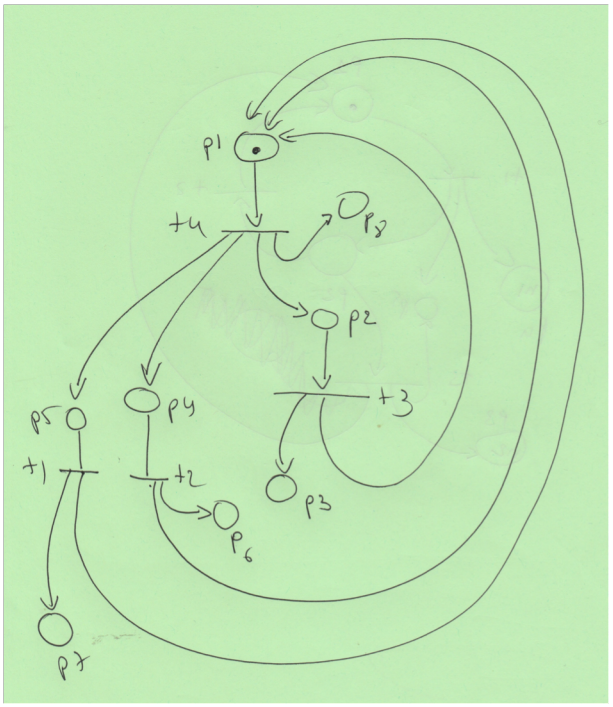

Figure 1: Réseau de Petri

Exercice 2 *Compléter le module pluscalappaspd11.tla en proposant une assertion P1 correcte.*

```
---------------------------- MODULE pluscalappaspd11 ----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*
--wf
--algorithm ex1{
variables x = 1;


process (one = 1)
{
  A:
    x := x - 1;
};
```

```
process (two = 2)
{
  C:
    x := x + 1;
  D:
    assert E1;
};

}
end algorithm;

*)

====
```

Exercice 3 *Compléter le module pluscalappaspd22.tla en proposant une assertion Q1 correcte.*

```
---------------------------- MODULE pluscalappaspd22 ----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*
--wf
--algorithm ex1{
variables x = 0;


process (one = 1)
variables u;
{
  A:
    u := x+1;
  AB:
    x := u;
  B:
    x := x +1;
};

process (two = 2)
{
  C:
    x := x - 1;
  D:
    assert E2;
};

}
end algorithm;

*)


====
```

Exercice 4 *Compléter le module pluscalappaspd33.tla en proposant deux assertions R1 et R2 correctes.*

```
---------------------------- MODULE pluscalappaspd33 ----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*
--wf
--algorithm ex3{
variables x = 0, y = 2;


process (one = 1)
variable u;
```

```
{
  A:
  u := x+1;
  AB:
    x := u;
  B:
    y := y -1;
  C:
 assert E31;
};

process (two = 2)
{
  D:
    x := x - 1;
  E:
    y:=y+2;
   F:
    x:= x+2;
  G:
    assert E32;
};

}
end algorithm;

*)
\
====
```