TD 1 — Prise en main de l'outil spin

Exercice 1 — Canaux

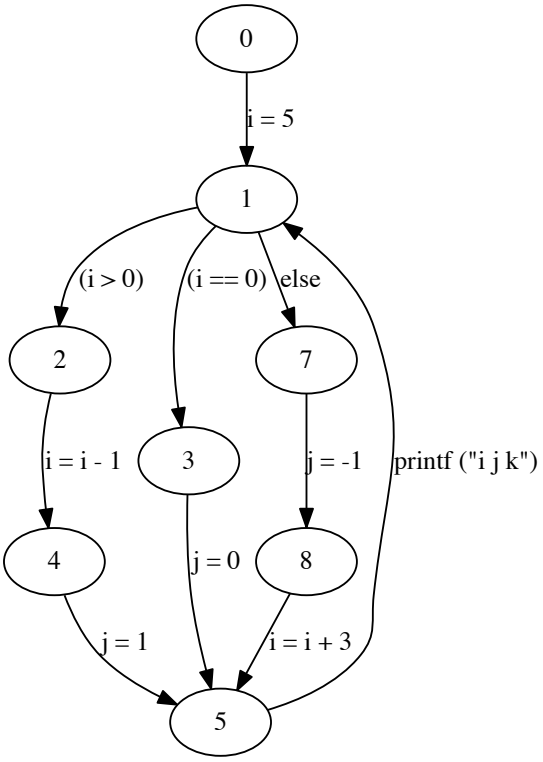
Écrire un programme Promela constitué de deux processus, p et q. Le processus p choisit, dans une boucle, un nombre entier x entre 0 et 5 puis il l'envoie au processus q. Ensuite il attends la réception d'un autre nombre y du processus q et il vérifie la validité de la condition $0 \leq y \leq 50$ à l'aide d'un `assert`. Le processus p termine sa boucle lors que y est égal à 0.

Le processus q reçoit des entiers du processus p, il les multiplie par 50 et les renvoie au processus p. Il s'arrête lors qu'il a reçu (et renvoyé) l'entier 0.

Q. 1.1 Écrire un programme Promela décrivant ce problème.
Q. 1.2 Varier la taille de/des canaux de communication. Varier aussi la taille de l'intervalle dans lequel x est choisit. La taille de l'espace d'états, comment évolue-t-elle ?

Exercice 2 — Rétro-ingénierie

Soit l'automate suivant :



Q. 2.1 Écrire un programme Promela tel que l'automate associé au programme soit identique (ou très similaire) à celui-ci.

Exercice 3 — Tirage au sort

On considère deux processus p et q. Chacun d'eux devra tirer de façon non-déterministe une valeur de 0 à 3 puis il devra échanger la valeur tirée avec l'autre (à l'aide d'un canal). Le processus qui aura tiré la valeur la plus grande sera déclaré vainqueur. Les deux processus s'arrêteront alors et le gagnant enverra au processus `init` la valeur qu'il a tirée ainsi que la valeur de l'autre processus.

Le processus `init` affichera ces valeurs avec l'identité du gagnant (p ou q). Si les deux valeurs tirées par p et q sont égales, ils devront recommencer l'opération.

Q. 3.1 Écrire un programme Promela décrivant ce problème.
Q. 3.2 Simuler ce programme.
Q. 3.3 Ajouter une assertion au processus `init` vérifiant que les valeurs envoyés par le processus gagnant sont différentes. Vérifier que l'assertion reste vrai pour tout état accessible.

Exercice 4 — Crible d'Eratosthène

Le crible d'Eratosthène est une méthode pour trouver les nombres premiers de 1 à N qui repose sur l'observation que n est premier si il n'y a aucun m premier plus petit que n tel que m divise n.

Un algorithme parallèle basé sur cette méthode consiste à créer une file de processus qui filteront des entiers reçus. Chaque processus est responsable d'un nombre premier n. Un processus reçoit des nombres du processus précédant dans la file. Si n divise un nombre reçu r alors r n'est pas premier et peut donc être ignoré. Dans le cas contraire, r est passé au processus suivant dans la file qui effectuera le même traitement. Si le processus était le dernier de la file alors r est un nombre premier et un nouveau processus responsable du nombre premier r est créé en fin de file. Initialement, la file est uniquement constituée du processus responsable du nombre premier 2. A tout moment, une file de k processus est donc constituée de processus responsables des k premiers nombres premiers.
Q. 4.1 Ecrire un programme Promela implémentant cette méthode à partir du squelette ci-dessous. Un processus filtre reçoit à sa création le nombre premier n dont il sera responsable ainsi que le canal *cin* sur lequel il recevra des entiers. Si il est le dernier de la file il pourra éventuellement créer un nouveau processus de filtre en lui passant un nouveau nombre premier r ainsi qu'un canal *cout* qu'il aura déclaré et sur lequel il enverra ensuite des entiers. Le processus `init` devra créer le processus de filtre reponsable du nombre 2 et lui envoyer ensuite tous les nombre de 3 à N.

```
#define N 20
proctype filtre(int n, chan cin) {
   chan cout = [5] of int;
   ...
   run filtre(r, cout);
   ...
}
init {
  chan c = [5] of int;
  run filtre(2, c);
  /* boucle pour envoyer sur c tous les nombres de 3 à N */
  ...
}
```

Q. 4.2 Simuler ce programme. Ajouter des printf pour afficher chaque nombre premier découvert.
Q. 4.3 Générer le graphe des états accessibles du programme.