TD 7 Esterel

Exercice 1 – Producteur-consommateurs : signaux valués multiples

Le problème assemble un producteur, deux consommateurs et le gérant du stock de l'unique produit : il met à jour le nombre de produits disponibles.

Les signaux d'entrée sont : C1 et C2 pour indiquer la demande d'un des deux consommateurs, arrivant aléatoirement de l'extérieur et FIN qui arrête le travail de la journée.

Dans un premier temps on accepte une seule demande par instant : écrire déclaration et relation.

On impose le comptage des produits reçus par chaque consommateur, le comptage des produits fabriqués par le producteur et évidemment le stock. Les variables `stock`, `nbprod`, `nbconso1` et `nbconso2` sont sorties en fin de programme grâce respectivement aux signaux de sortie valués `FS`, `FP`, `FC1` et `FC2`.

Question 1

Donner la structure générale du programme.

Question 2

Ecrire le producteur : il produit dès le départ, émettant le signal de sortie `P` après 3 tick (destiné au gérant), incrémente sa production et attend pour recommencer à produire le signal P_S (produit stocké) émis par le gérant.

Question 3

Un consommateur recevant l'entrée qui lui est associée (C1 ou C2) envoie un signal C valué par son numéro : il demande ainsi un produit au gérant et attend C_S(num) (consommateur servi) avant de reprendre toute commande. Lorsque le signal C_S(num) le concerne, il incrémente le nombre de produits reçus. Une demande C1 n'est pas nécessairement satisfaite, le gérant répond alors par C_S(0).Ecrire le consommateur 1.

Question 4

Le gérant est plus complexe. On impose que le traitement des signaux du producteur et du consommateur soit réalisé par des branches parallèles : `stock` ne peut être partagée -même en lectuire seule- et doit être consultée et modifiée dans un traitement unique. Pour cela, le signal multiple valué `S` peut être émis plusieurs fois pour indiquer les modifications souhaitées par les branches (+1 pour accepter un produit et -1 pour livrer un produit) ; il présente le bilan de traitement des branches et permet la mise à jour du stock en fin d'instant. De la même manière, des signaux internes, locaux au gérant, `MAX_L` et `VIDE_L` émis permettent d'avertir les branches des conditions de blocage temporaire de leur activité.

Le point délicat concerne la reprise de la production lorsque le stock est plein. Rappel : sauf pour les signaux multiples, un *signal d'entrée* deja présent dans l'instant ne peut y être émis de nouveau.

Exercice 2 – Calculs en C appelés dans esterel

Esterel n'est pas limite à l'émisson de signaux : par exemple, des valeurs de régulation à transmettre peuvent faire appel à des calculs complexes, appelant des bibliothèques de calcul scientifique C. L'appel d'une fonction C se fait par l'instruction call fonction(liste param résultat)(liste param d'entrée). La liaison est évidemment facilitée par le fait que le traducteur esterel produit du C et l'incorporation de procédures se fait facilement lors de l'édition de lien. Cependant, cette étape doit être soigneusement préparée, d'une part par une déclaration dans le code esterel séparant en deux listes distinctes les paramètres de retour et les paramètres d'entrée (dans cet ordre) et d'autre part dans un fichier nom.h (syntaxe C) portant le même nom que le programme esterel nom.strl. Les paramètres ont des valeurs simples (entiers, flottants, chaines, booléens) ; dans la fonctions C associée, les données d'entrée sont des valeurs et les résultats sont des pointeurs sur les types correspondant. Le premier exercice émet simplement des signaux à valeurs aléatoires entre 1 et une valeur maximum. Voici le programme C permettant l'initialisation du generateur et la production d'une nouvelle valeur entre 0 et borne - 1 :


```
#include <stdio.h>
#include <sys/types.h>
#include <time.h>
#include <limits.h>
#include <stdlib.h>

void init (int *sortieinutile, int entreeinutile) {
  time_t t1;

  (void) time(&t1);
  srand((long) t1);
}

void alea (int *resalea, int borne) {
  int rd;

  rd = rand();
  *resalea = (((double) rd) / RAND_MAX) * borne;
}
```

Question 1

Ecrire le programme `delaialea.strl` comportant une boucle qui, lors de chaque pas :

- demande le calcul d'un nouveau nombre entre 0 et une borne (- 1) donnée,
- émet la valeur obtenue dans un signal de sortie valué S,
- utilise ce nombre pour faire une attente de 1 à borne `tick`.

Deuxième exemple : On veut simuler avec cette technique l'arrivée aléatoire de 3 clients numérotés de 1 à 3 : une branche demande le calcul d'un nombre et émet la valeur obtenue vers les autres branches au travers d'un signal valué. Trois autres branches comptent simplement le nombre d'arrivée de chacun des clients. Un signal `FIN` arrête le processus et provoque l'affichage des trois compteurs.

Question 2

Ecrire une version formée de 4 boucles se déroulant en parallèle.

`alea.c` contient les definitions de alea et init
clientalea.h doit contenir les declarations c des fonctions init et alea
gcc -ansi -Wall -c alea.c ,pour creer alea.o
esterel -simul clientalea.strl cree clientalea.c comme d'habitude
gcc -o clientalea clientalea.c alea.o ../libcsimul.a cree clientalea, pret a executer ...
avec sequence `;;;;;;;;;;;;;;;;;;;;;;;...;;FIN;`

exemple de sortie : F1(446) F2(454) F3(467)

Question 3

On peut remarquer que chaque boucle exécute exactement un pas lors de chaque instant ; on peut donc écrire une seule boucle contenat des braches parallèles. Modifier le programme précédent pour mettre en uvre cette solution.

Exercice 3 – Une machine à café modulaire

On veut écrire un module attend_prix réutilisable dans de mutiples distributeurs pour recevoir une certaine somme fixée par une constante prix_produit, 4 par exemple. Dans un premier temps on accepte uniquement des pièces de 1, 2 unités monétaires (signal valué d'entrée PIECE, de sortie PIECE_REJETEE pour une autre pièce). Le module cumule les entrées jusqu'à ce que le prix soit atteint ou dépassé, tout en surveillant l'annulation (signal d'entrée

ANNULER) qui renvoie les pièces déjà émises (signal de sortie valué RETOUR).
Lorsque la somme cumulée atteint -ou dépasse le prix de la boisson, le module émet PRIX_ATTEINT ; si la somme cumulée dépasse le prix le module rend la monnaie (signal de sortie valué MONNAIE).
Il attend que la boisson soit prise (signal PRODUIT_SERVI) ou pour reprendre au début, attendre de nouvelles pièces.

L'instruction trap (condition portant sur des variables) permet de sortir de la phase d'acquisition des pièces et de rendre la monnaie (signal de sortie valué MONNAIE) si nécessaire ; l'instruction abort (portant sur les signaux) permet de traiter le retour de la somme déjà acquise (signal de sortie valué RETOUR).

Question 1
Ecrire l'ensemble du module.

Question 2
Le second module servir_produit est plus simple : il reçoit PRIX_ATTEINT, NUM_PRODUIT et émet SERVIR avec ce numéro du produit (signal de sortie valué NUM_PRODUIT) ; une pause de durée déterminée par la constante durée simule la préparation/livraison paramétrable et émet PRODUIT_SERVI.

Les signaux de communication entre les deux modules sont donc PRIX_ATTEINT et PRODUIT_SERVI.

Question 3
Ecrire une machine à boissons chaudes mach_boisson_run (on peut supposer que le numéro 1 est associé à un café, 2 à un thé, 3 à un chocolat) qui utilise ces modules. Veiller à ce que les signaux circulant (émis par l'un utilisé par l'autre) entre les modules soient internes au module de la machine. Les signaux entrants et sortants doivent être redéclarés dans la machine avec une correspondance assurée dans l'instruction run.Donner une séquence de signaux d'entrée pour tester le fonctionnement.

Exercice 4 – Lecteur de CD lectcd lectcdpar

On veut simuler un lecteur de CD, sans fonction pause : on arrête la musique soit en ouvrant le tiroir, soit parce que le signal FIN_CD arrive de l'extérieur.*Dans cette première version, on impose un mouvement du tiroir entre le retrait d'un disque et le positionnement d'un nouveau disque.*Les signaux d'entrée :TIROIR (bascule ouvrant et fermant le tiroir), prioritaire arrête le sonPLAY démarre le sonMCD et PCD indiquent la mise (ou prise) d'un CD par l'utilisateur. FIN_CD indique la fin normale du disqueLes signaux de sortie :Musique, émis lors de chaque instant lorsque PLAY est appuyé dans l'état 'tiroir fermé et plein' SansDisque, lorsque PLAY est appuyé dans l'état 'tiroir fermé et vide', auxquels s'ajoutent des signaux échos destinés à la période de test :OUV_S, FERM_S (pour les deux changements d'état de TIROIR), MCD_S, PCD_S, FIN_S, CD_S(boolean) informe également sur la présence d'un disque dans le tiroir lors de chaque mouvement.Deux variables booléennes disque et ouvert conservent les informations d'état nécessaires entre les signaux.

Question 1
Quel est le signal principal, prenant le pas sur tous les autres ? Quels sont les états important (le mot état est pris ici au sens usuel : situation stable changée par certains signaux) ?

Question 2
Ecrire un programme séquentiel décrivant un lecteur de CD.

every Send every ***équivaut à (mieux vaut utiliser la 2eme forme déjà vue)****loopabortwhen Send loop*

Question 3
On veut écrire une version parallèle séparant :
- la gestion spécifique du signal des mouvements du tiroir
- la gestion des signaux MCD / PCD
- la gestion du son par PLAY / FIN_CD

Les portions de code de la version séquentielle concernant ces parties ne peuvent être mises en parallèle puisqu'elles partagent les variables.Deux solutions classiques peuvent être mises en oeuvre :

— un changement d'état déclenche un signal 'continu' (émis à chaque instant d'un état stable hors abandon) pour informer les autres branches ;
— une variable partagée dont la valeur ne change pas à chaque instant est dupliquée à un point de synchronisation sans parallèlisme ; les branches travaillent alors sur l'état connu en fin d'instant précédent encore valide (puisque que le signal provoquant le changement de valeur est prioritaire).

On utilise la première technique pour la variable ouvert, la seconde pour la variable disque dupliquée en une variable cd utilisée par la branche traitant PLAY. Ecrire le module correspondant.

Exercice 5 – Threads concurrents et synchronisation en Esterel

On veut simuler le rayon `fruits` d'une supérette : on suppose qu'il y a trois étalages de fruits (Bananes, Pommes, Oranges), initialement fournis chacun avec 10 kilos de fruits et au plus deux clients qui se servent simultanément dans le même bac. Un gérant automatique de chaque étalage connaît à chaque instant la quantité disponible de chaque fruit, il doit prévenir un employé lorsqu'il reste moins de 2 kilos d'un fruit pour qu'un cageot soit ajouté et éviter ainsi la rupture de stock. Le signal d'entrée `B1, O1`, ou `P1` (resp. `B2, O2`, ou `P2`) indique que le client 1 (resp client 2) souhaite se servir un kilo du fruit B, O ou P. Bien sur, un client ne peut se servir que d'une sorte de fruit à la fois. Un instant plus tard, le client dispose de son kilo de fruit, chaque balance intelligente émet l'un des signaux valués `B(1), O(1), P(1)` (destiné au gérant automatique) ; bien sûr, plusieurs clients peuvent émettre ce signal au même instant. Le gérant des bananes émet le signal `MB` pour indiquer qu'il risque de manquer de bananes lorsqu'il en reste 2 kilos ou moins, celui des oranges émet `MO`, celui des pommes `MP` dans les mêmes circonstances. Plusieurs de ces signaux peuvent être émis au même instant. Il reçoit un signal interne (ou déclaré output pour être observable en phase de test) AB (resp. AO ou AP) lorsque 10 kilos de bananes (resp. oranges, pommes) ont été ajoutés à l'étalage. Ces signaux sont émis par l'employé lorsqu'il a rechargé l'étalage.

Question 1 – Déclarations et structure générale

Déclarer les signaux nécessaires. Indiquer l'organisation générale du programme : montrer tous les blocs parallèles (en indiquant leur rôle par une ligne de commentaire).

Question 2 – Les clients (en supposant qu'un étalage n'est jamais vide)

Ecrire les deux clients. Quelles sorties obtient-on avec la séquence d'ensembles d'entrées O1 ;B1 O2 ; P1 ; ?
Proposer une séquence d'ensembles d'entrées en incluant des instants où aucun client n'est présent au rayon fruit.

Question 3 – Les gérants

Déclarer les variables nécessaires. Ecrire le gérant des oranges. Proposer une séquence d'ensembles d'entrées menant à l'émission des signaux MO puis AO

Question 4 – L'employé, récepteur des signaux MB, MO, MP et émetteur de AB, AO, AP

Ecrire l'employé, en supposant qu'il peut recharger plusieurs étalages dans le même instant. Si on lève cette hypothèse, un étalage peut se trouver vide ; quelles modifications proposez-vous ?

Question 5

Les gérants sont complètement superposables sauf par le nom des signaux qu'ils échangent avec l'extérieur. Proposer une écriture du module gérant et trois instanciations dans le programme initial.

Exercice 6 – Robot NXT et Esterel

Nous avons adapté la sortie du compilateur `esterel` pour pouvoir la faire tourner sur une brique Mindstorm NXT.

Une brique mindstorm NXT est un microcontroleur (en fait 2 !) sur lequel nous avons installe une machine virtuelle java nommée `lejos`. Ce microntontroleur dispose des entrée suivantes :

— Boutons `ENTER,ESCAPE,LEFT,RIGHT`

— Un capteur de contact `Touch`
— Un capteur Ultrason `Distance`
— Un cpateur de Son `Sound`
— Un capteur de Lumiere, interrogé avec `Depassement_Lumiere` pour le seuillage et `Light` pour sa valeur courante.

Comme sortie nous disposons des sorties suivantes :
— `A_Avance,B_Avance,C_Avance`
— `A_Recul,B_Recul,C_Recul`
— `A_Stop,B_Stop,C_Stop`
— `A_Vitesse,B_Vitesse,C_Vitesse`
— `Bip`
— `LCD, LCDint, LCDfloat` (affichage de string integer et float respectivement).

De part l'existence physique de ces différents objets, nous avons une contrainte quant à l'interface minimale requise :

```
input ENTER,ESCAPE,LEFT,RIGHT;
input A_isMoving,B_isMoving,C_isMoving ;
input Distance : integer,Depassement_Lumiere, Sound;
input Touch;

output A_Vitesse : integer, C_Vitesse : integer;
output A_Avance, C_Avance,tmpsig,LCD: string;
output LCDint : integer, LCDfloat : float;
output A_Recul, C_Recul;
output A_Stop, C_Stop;
```

Question 1
En quoi `Light` est il un sensor au sens esterel du terme mais `Touch` un simple signal d'entrée ?

Question 2
Ecrire un programme esterel attendant le signal `Touch` et émettant le signal `Bip`.

Une fois ce programme écrit Pour le transferer sur la briquer NXT le protocole est le suivant :
— compiler le programme vers java à l'aide du script `doall`
— Compiler le java à l'aide du compilateur `nxjc`
— Effectuer l'edition de liens, commande `nxjlink`.
— Charger le binaire produit dans le robot avec la commande `nxjupload`.

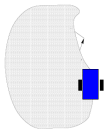
Ces deux dernieres étapes peuvent être fusionnées en une seule en utilisant la commande `nxj`.

Question 3
Ecrire un programme esterel avancant le robot pendant 30 instants puis reculant le robot pendant 30 instants puis ravancant etc... Combien de secondes durent 30 instants ? Ici vous devrez enlever les piles du robot pour l'arreter.

Question 4
Modifier le programme précédant pour que l'appui sur `ESCAPE` mette fin au programme. A partir de maintenant il sera de bon ton de toujours implanter ce mecanisme dans vos programmes !

Question 5
Ecrire un programme permettant de calibrer le capteur de lumière pour qu'il sache distinguer une zone claire d'une zone foncée. On procédera comme suit : on placera le robot sur une zone claire on appuiera sur le bouton `ENTER` puis on le placera sur une zonne foncée et on recommencera. On fera finalement afficher les deux valeurs du *sensor* `Light`.

Question 6
Ecrire un programme de suivi de forme : on souhaite que le robot utilise son capteur de lumiere pour detecter à quel moment le robot chevauche une zone claire et une zone foncée. Lorsque lerobot detecte un changement de surface, il se met a tourner sur place (bloque une roue) pour que le capteur se repalce a nmouveau au dessus de l'ancienne surface, puis il bloque lautre roue et se met a tourner dans l'autre sens pour replacer le capteur au dessus ede la nouvelle surface etc.... ce qui nous donne le schema suivant :



Question 7
Ecrire un programme qui met le robot à distance constante d'une cible mobile placée devant le robot.

Question 8
Modifier le programme precedant pour que le passage du robot sur un zone foncée provoque l'emission d'un Bip.

Question 9
Ecrire un programme avancant le robot en ligne droite et lorsque qu'un obstacle surgit sur la route le robot le contourne et reprend sa trajectoire initiale.

Question 10 – Question bonus
Ecrire une Tache java mémorisant l'itinéraire fait par le robot puis lorsque le bouton entre est appuyé effectue l'itinéraire en sens inverse (on aura besoin d'un signal d'input et d'un signal d'output supplémentaire....)

