Programmation Synchrone — M2 Informatique et EIDD Université Paris Diderot

Examen du 13 décembre 2016

Barème indicatif. Documents interdits. Toute la programmation se fait en SCADE
Durée : 3h (de 15h30 à 18h30). Amphi: 3B.

Exercice 1 (3 pts) – *Terminologie*

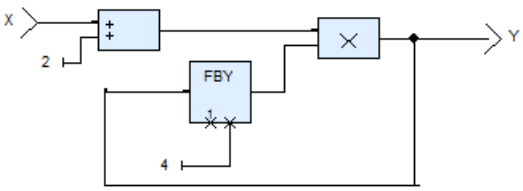
Expliquez en 1-3 lignes chacune des notions suivantes. *Barème avec bonus-malus : réponse fausse -0.25 ; pas de réponse 0 ; réponse incomplète ou imprécise 0,25 ; bonne réponse 0,5.*

1. Logiciel critique
2. Validation par revue de code.
3. Hypothèse synchrone.
4. Logiciel de temps réel dur.
5. Causalité.
6. Cycle de développement en Y.

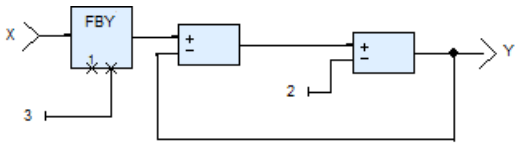
Exercice 2 (2pts) – *Modèle synchrone*

Est-ce que les opérateurs Scade ci-dessous sont corrects ? Si c'est le cas, quelle est leur sortie Y pour l'entrée $X = 1, 2, 3, 2, 1, 0 \ldots$. Sinon expliquez pourquoi.

1. Noeud A



2. Noeud B



Exercice 3 (2 pts) – *Vérification*

Étant donné un opérateur `toto` avec les entrées réelle `m,n` et une sortie booléenne `alarme`, comment peut-on démontrer avec Scade chacune des propriétés suivantes :

1. si $m > n$ pendant 10 ticks, alors l'alarme sonne ;
2. si toujours $m > n$, alors à chaque instant où $m = 10$. l'alarme sonne.

Exercice 4 (9 pts) – *Ascenseur*

Dans ce problème vous devez programmer la commande d'un ascenseur. La note dépendra de la qualité du code (modularité, clarté). Les ticks de l'horloge ont lieu toutes les secondes.

1. L'ascenseur simple se trouve à l'altitude `z` (mesurée par un capteur) et doit arriver à l'altitude `zSouhaitee`. Pour cela il monte (s'il est trop bas) ou il descend (s'il est trop haut). Pour monter, il doit envoyer 1 au moteur, pour descendre -1, et 0 pour s'arrêter.
   Programmez un noeud Scade pour un tel ascenseur, avec 2 entrées réelles `z` et `zSouhaitee`, et une sortie entière `moteur`.

2. Améliorez votre ascenseur pour qu'il attende 20s, puis monte jusqu'à la position souhaitée, et là s'arrête encore pour 20 secondes.

   Programmez le noeud Scade correspondant (mêmes entrées-sorties que dans le point 1).

3. On est maintenant dans un immeuble avec un rez-de-chaussée et 10 étages, leurs altitudes sont `zEtage[0],zEtage[1],zEtage[2],...,zEtage[10]`. A chaque étage se trouve un bouton dont l'état est représenté par un booléen `bEtage[0],bEtage[1],bEtage[2],...,bEtage[10]`.

   Programmez un "décodeur" qui a deux entrées : `zEtage: real^11` et `bEtage: bool^11` et deux sorties `zSouhaitee: real` et `bouton: bool`. Quand un bouton d'un étage est appuyé, la sortie `bouton` devient vraie et la sortie `zSouhaitee` prend la valeur de l'altitude de cet étage. Si le bouton est relâché, la sortie `bouton` devient fausse, et la sortie `zSouhaitee` garde sa valeur.

   Expliquez en deux lignes : (1) comment la sortie `zSouhaitee` est initialisée ; (2) que fera votre décodeur si deux (ou plus) boutons sont appuyés au même moment.

4. **Modifiez votre décodeur** pour qu'il tienne aussi compte des boutons de la cabine `bCab[0],bCab[1],bCab[2],...,bCab[10]`.

5. Si on connecte votre décodeur (points 3-4) à l'ascenseur de point 2, on obtient un ascenseur fonctionnel, mais pas parfait. En particulier, si on appuie un bouton pendant que la cabine fait une course, elle change de destination tout de suite.

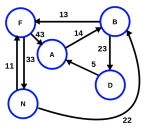
   Pour remédier à ça, **faites de sorte que** l'ascenseur puisse être libre ou occupé. Quand il est libre il accepte le premier appel venu, passe à l'état occupé, va à l'étage souhaité et revient à l'état libre. Dans l'état occupé il ne réagit pas aux appels.

Exercice 5 (4 pts) – *Ordre supérieur*

1. **Facile — intégration** Étant donné un opérateur `f` (avec une entrée et une sortie réelles) qui calcule une fonction $f : \mathbb{R} \to \mathbb{R}$, programmez un noeud `intf` (sans entrée et avec une sortie réelle `iii`) qui calcule $\int_0^1 f(x)\,dx$ par la formule de trapèzes :

$$iii = \int_0^1 f(x)\,dx \approx (f(0)/2 + f(0,01) + f(0,02) + \cdots + f(0,98) + f(0,99) + f(1)/2) \cdot 0.01$$

2. **Difficile — Bellman-Ford** Le graphe pondéré G de n sommets (numérotés de 0 à $n-1$) est donné par sa matrice de poids W (de taille $n \times n$, et telle que W_{ij} est le poids de l'arc (i,j)). On souhaite calculer les distances (poids minimum des chemins) à partir du sommet 0 vers chacun des sommets i, le tableau (de taille n) obtenu s'appelera D. On vous rappelle l'algorithme de Bellman-Ford en pseudocode :




```
relax(i,j)
    D[j]=min(D[j], D[i]+W[i,j])

init(D)
    D[0]=0
        pour i de 1 a n-1
            D[i]=infini
```

```
BF(W)
    init(D)
    repeter n fois
        pour tout i
            pour tout j
                relax(i,j)
    retourner D
```

Programmez un opérateur Scade (avec entrée W et sortie D) qui implémente cet algorithme.