LICENCE SCIENCES & TECHNOLOGIES
1re ANNÉE

UE INF201, INF231
ALGORITHMIQUE ET PROGRAMMATION FONCTIONNELLE

2020 / 2021

DEVOIR SURVEILLÉ — mars 2020

- Épreuve de 2 heures ; document autorisé : une feuille A4 manuscrite recto-verso ; appareils électroniques interdits (à l'exception d'une montre).
- Le sujet comporte un problème et un questionnaire à choix multiples. L'ensemble est noté sur 45 points. Le barème est donné à titre indicatif.
- Si vous êtes bloqué par une question, passez à la suivante qui peut être plus facile.

  Dans toute question, il est possible d'utiliser une fonction d'une question précédente sans l'avoir définie. En revanche, **l'utilisation d'une fonction auxiliaire de votre invention n'est possible qu'après l'avoir spécifiée et réalisée**.
- Prenez le temps de bien lire le sujet et de repérer les questions faciles.

Table des matières

1 Problème : les ronds-points, y'en a assez !

25 pts

Vous êtes chargé par l'entreprise dans laquelle vous travaillez de concevoir le prototype d'un système révolutionnaire de pilotage de feux quadricolores pour les carrefours à trois branches, comme celui de la figure 1 ci-dessous :

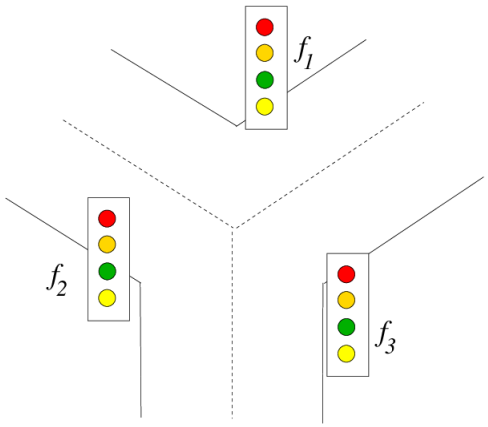

FIG. 1: feux dans un carrefour à trois branches

La sécurité routière a déjà eu des ennuis avec des systèmes informatiques défaillants qui ont causé des accidents. Elle exige donc que le système soit sans faille. En cas de panne ou d'accident, votre entreprise sera considérée comme responsable, pourra être attaquée en justice et devra payer les dommages et intérêts aux plaignants[1].

Pour s'assurer que le prototype envisagé est correct, nous allons modéliser les feux quadricolores ainsi que des fonctions de vérification de leur bon fonctionnement.

1.1 Principe de fonctionnement d'un feu quadricolore

Chaque feu possède quatre lampes de couleur respective jaune, verte, orange et rouge qu'il allume consécutivement dans l'ordre suivant :

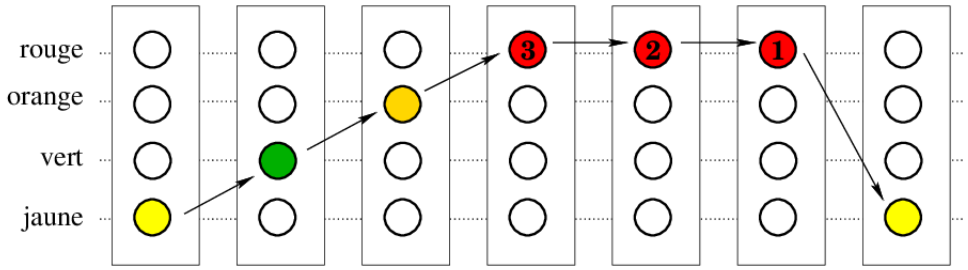

FIG. 2: cycle standard des feux

jaune : indique aux conducteurs de se préparer car le feu va bientôt passer au vert.
Il est interdit de traverser le carrefour quand le feu est jaune.

vert : indique qu'on peut traverser le carrefour.

orange : indique qu'il faut ralentir et se préparer à s'arrêter car le feu va bientôt passer au rouge.
On peut encore traverser le carrefour quand le feu est orange.

1. et vous serez viré.

rouge : indique qu'il est interdit de passer.

Le feu reste rouge pendant un certain nombre de minutes qui s'affiche sur la lampe rouge.

Dans la suite du problème, on décide que chaque feu reste rouge pendant 3 minutes décomptées de 3 à 1 comme sur la figure 2.

Le cycle décrit ci-dessus est appelé *cycle standard*.

Étant donnés l'ensemble des entiers naturels 1, 2 et 3 appelé *nat123*, et les quatre constructeurs J, V, O et R de profil suivant, on définit l'ensemble *sign* des signaux affichés par les feux :

- $J, V, O : sign$
- $R : nat123 \to sign$

$$sign \stackrel{def}{=} \{J, V, O\} \cup \{R(n) \text{ tel que } n \in nat123\}$$

Q1. (1.5pt) Donner les six représentations des signaux correspondant au cycle standard d'un feu (de jaune à rouge, revoir la figure 2) avec un feu rouge qui indique les minutes restantes de 3 jusqu'à 1.

Q2. (1.5pt) Implémenter *nat123* (on veillera à préciser les contraintes éventuelles) et *sign*.

On spécifie une fonction permettant de passer au signal suivant du cycle :

Profil $sig\_suiv : sign \to sign$
Sémantique : *$sig\_suiv(s)$ est le signal qui suit le signal s dans le cycle standard.*

Q3. (1pt) Compléter les exemples d'utilisation de *sig_suiv* :

1. $sig\_suiv(O) = \ldots\ldots\ldots\ldots$
2. $sig\_suiv(R(1)) = \ldots\ldots\ldots\ldots$
3. $sig\_suiv(R(n)) = \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

Pour le 3e exemple, préciser les valeurs de n pour lesquelles l'équation est valide.

Q4. (2pt) Implémenter *sig_suiv*.

Deux feux ont des signaux dits *compatibles* si et seulement si l'affichage simultané de ces signaux n'entraîne pas un risque d'accident. On spécifie ainsi un prédicat (fonction de codomaine $\mathbb{B}$) assurant la compatibilité de deux signaux :

Profil $sig\_compat : sign \to sign \to \mathbb{B}$
Sémantique : *Les seuls cas incompatibles sont les signaux indiquant tous les deux aux conducteurs qu'ils peuvent traverser le carrefour (relire l'introduction de cette section).*

Q5. (1pt) Quels sont les cas incompatibles ?

Q6. (1.25pt) Compléter les exemples et la propriété :

1. V et V sont $\ldots\ldots\ldots\ldots$; V et J sont $\ldots\ldots\ldots\ldots$
2. $\ldots\ldots\ldots$ et $R(2)$ sont compatibles.

3. O et O sont .....

4. $\forall s \in sign,\ \forall n \in nat123$ ..... et s sont compatibles.

Q7. (2pt) Implémenter *sig_compat*.

1.2 Configurations du carrefour à trois branches

À un instant donné, la configuration des feux f_1, f_2 et f_3 d'un carrefour à trois branches comme celui de la figure 1 sera modélisée par un triplet de signaux.

Par exemple, la configuration (V, O, J) indique que le feu f_1 est vert, le feu f_2 est orange et le feu f_3 est jaune.

Q8. (0.75pt) Implémenter un type appelé `config` représentant les configurations des carrefours à trois branches.

Une configuration est dite *sans danger* pour les conducteurs si les signaux des trois feux sont compatibles deux à deux. On spécifie ainsi le prédicat suivant :

Profil $configSD : config \to \mathbb{B}$
Sémantique : *$configSD\ (c)$ est vrai si et seulement si la configuration s est sans danger.*

Q9. (1pt) Donner quatre exemples distincts d'utilisation de ce prédicat : 3 avec une configuration dangereuse et 1 avec une configuration sans danger ; tous les exemples doivent inclure le constructeur *R*.

Q10. (2pt) Implémenter *configSD*. L'utilisation de la composition conditionnelle et de l'analyse par cas par filtrage est interdite dans cette question ; votre réponse ne devra donc contenir ni `if then else`, ni `match`.

On considère maintenant des séquences de configurations :

```
type seqconfig = V | C of config * seqconfig
```

`V` construit des séquences vides ; `C` construit des séquences non vides par ajout à gauche d'une configuration sur une séquence de configurations.

On étend aux séquences la notion de configuration sans danger :

Profil $seqconfigSD : seqconfig \to \mathbb{B}$
Sémantique : *$seqconfigSD\ (s)$ est vrai si et seulement si toutes les configurations de la séquence s sont sans danger.*

Q11. (1.5pt) Donner des équations définissant *seqconfigSD*. L'implémentation en OCaml n'est pas demandée dans cette question.

Q12. (1.5pt) Définir une mesure et montrer que $\forall s \in seqconfig$, l'évaluation de $seqconfigSD(s)$ termine.

Q13. (2pt) Implémenter *seqconfigSD*.

1.3 Changements de configuration

Pour vérifier le bon fonctionnement du carrefour à trois feux, on veut s'assurer que les feux respectent les contraintes suivantes lorqu'ils changent de configuration :

contrainte 1 : au moins un feu est passé au signal suivant (par un changement de couleur ou bien par un changement de minute) ;

contrainte 2 : chaque feu a soit conservé le même signal, soit est passé au signal suivant.

Autrement dit, on veut éviter que des feux ne changent jamais (contrainte 1) et que des feux changent de signaux sans respecter le cycle standard de la figure 2 (contrainte 2).

Un enchaînement de configurations respectant ces contraintes est dit *correct* :

Profil $enchainOK : config \to config \to \mathbb{B}$
Sémantique : $(enchainOK\ c_1\ c_2)$ = *vrai si et seulement si l'enchaînement de la configuration* c_1 *à la configuration suivante* c_2 *respecte les contraintes 1 et 2 détaillées ci-dessus.*

Q14. (2pt) Compléter les exemples d'utilisation de *enchainOK* en <u>justifiant</u> vos réponses :

1. ($enchainOK\ (J,V,R(1))\ (J,V,R(1))$) = ... car .....
2. ($enchainOK\ (J,V,R(2))\ (J,V,R(1))$) = ... car .....
3. ($enchainOK\ (J,V,R(2))\ (V,O,R(1))$) = ... car .....
4. ($enchainOK\ (J,V,R(1))\ (J,O,V)$) = ... car .....

Q15. (2pt) Implémenter *enchainOK* .

On étend la notion d'enchaînement correct aux séquences de configuration grâce au prédicat suivant :

Profil $seqconfigOK : seqconfig \to \mathbb{B}$
Sémantique : $seqconfigOK\ (s)$ *est vrai si et seulement si tous les enchaînements des configurations de s sont corrects.*

Q16. (2pt) Donner au choix des équations ou une implémentation de *seqconfigOK* .

2 Questionnaire à choix multiples

20 pts

NE PAS RÉPONDRE SUR CET ÉNONCÉ.
UTILISEZ LA FEUILLE DE RÉPONSES.

Questionnaire du QCM d'INF201, INF231

Répondre sur la feuille réponse à part

Question 1 ♣ Quelle est le type de l'expression `char_of_int(int_of_char('a')+1);;`

A `char`
B `int`
C `float`
D `bool`

Question 2 ♣ Quel est le type de la fonction `f` définie par
`let f (x:int) (y:int):  int = (x+y)/2`

A `float`
B `int→int`
C `int→int→int`
D `int`

Question 3 ♣ L'expression `2=(3=false)`

A s'évalue à `true`
B s'évalue à `false`
C renvoie une erreur de type

Question 4 ♣ L'expression `if a then a else not a` est équivalente à

A `not a`
B ne peut pas s'exprimer simplement avec des opérateurs logiques.
C `false`
D `true`

Question 5 ♣ L'expression `if a then b else (c=a)` est équivalente à

A if a then b else not c
B `a || b || c`
C `(a || c) && ((not a) || b)`
D `a || (not b) || c`
E aucune des autres réponses.
F `(a && b) || ((not a) && (not c))`
G if a then b else true
H `(not a) || b`

Question 6 ♣ On veut créer un type `etd` qui renseigne pour un étudiant donné son âge, son numéro d'étudiant, son prénom et son nom. Quelle est la solution à choisir parmi les suivantes:

A
```
type etd = int * int * string * string
(*dans l'ordre (age, numéro d'étudiant, prénom, nom)*);;
```
B `type etd = Age(int) * Num_etd(int) * Prenom(string) * Nom(string);;`
C `type etd = Age | Num_etd | Prenom | Nom;;`
D `type etd = Age of int | Num_etd of int | Prenom of string | Nom of string;;`

Question 7 ♣ On veut créer un type `distance` qui permet de donner des distances en kilomètres ou en miles. Quelle sont les solutions envisageables parmi les suivantes:

A `type distance = float * float (*dans l'ordre (km, miles)*);;`
B `type distance = Miles of float | Km of float;;`
C `type distance = float*unite_dist;;` où le type `unite_dist` a été défini auparavant par `type unite_dist= Km|Miles;;`
D `type distance = Km * Miles;;`

Question 8 ♣ Si x et y n'ont pas encore été définis, la valeur de l'expression `let x = 10 in let x = 3+x in let y = x+4 in x+y;;` est

- A inexistante (erreur)
- B `17`
- C `7`
- D `3`
- E `24`
- F `30`
- G `10`

Question 9 ♣ On définit la fonction `f` par `let f b x y = if b then x=y else x;;`

- A `x` et `y` sont forcément de même type.
- B `x` et `y` sont forcément de type `bool`.
- C `(f true true true)` vaut `true`
- D `(f false 0 5)` vaut `0`
- E `(f true 0 5)` vaut `false`
- F `b` est de type `bool`
- G `(f true 5 5)` vaut `true`

Question 10 ♣ On définit
`let rec u n = if n=0 then 0 else 2*(u (n-1));;`.
A quoi est égale `(u 3)`:

- A 30
- B 16
- C 10
- D 14
- E 12
- F 0
- G 8
- H 28

Question 11 ♣ Soient `f` et `g` définies par `let f x = 3*x and g y = y+4;;`. Quelle est la valeur de `(g 6)*(f 2)`

- A 42
- B (6,10)
- C 60
- D 31
- E 2020
- F (10,6)

Question 12 ♣ On définit la fonction `compare` par
`let compare (bi1,bs1 :int*int) (bi2,bs2 :int*int) : bool =`
`(bi1<bi2 && bs1<bs2)`
Parmi les expressions suivantes lesquelles valent `true`.

- A `compare (2,5) (2,8)`
- B `compare (4,5) (2,3)`
- C `compare (2,3) (1,5)`
- D `compare (1,5) (2,6)`
- E `compare (8,3) (10,5)`
- F `compare (0,2) (2,3)`

Question 13 ♣ On veut implémenter $(x, n) \mapsto x^n$ avec x et n entiers positifs ou nuls. Parmi les implémentations suivante, lesquelles sont correctes:

- A
```
let rec power ((x,n) :int*int):int=
x * (power (x,n-1));;
```
- B
```
let rec power ((x,n) :int*int):int=
match n with
|0->1
|n-> if (n mod 2)=0 then
       (power (x*x,n/2))
     else
       x*(power (x*x,(n-1)/2));;
```
- C
```
let rec power ((x,n) :int*int):int=
match n with
|0->1
|n-> x*(power (x,n-1));;
```
- D
```
let rec power ((x,n) :int*int):int=
match n with
|0->1
|n-> if (n mod 2)=0 then
   (power (x*x,n/2))
|n-> if (n mod 2)=1 then
    x*(power (x*x,(n-1)/2));;
```
- E
```
let rec power ((x,n) :int*int):int=
match n with
|n-> x*(power (x,n-1))
|0->1;;
```
- F
```
let rec power ((x,n) :int*int):int=
if n=0 then 1 else x * (power (x,n-1));;
```

Question 14 ♣ On définit le type `intlist` par `type intlist = Nil | Cons of int*intlist;;`
On définit la fonction `f` par

```
let rec f (l:intlist):intlist = match l with
|Cons(x,Cons(y,ll))-> Cons(x,Cons(x, f ll))
|l-> l
```

A `f` termine pour toute entrée

B `f (Cons(3,Cons(2,Cons(1,Nil))))` renvoie une erreur

C `f (Cons(2,Cons(1,Nil)))` vaut `Cons(2,Cons(2,Nil))`

D Le filtrage (pattern matching) n'est pas exhaustif.

E `f (Cons(3,Cons(2,Cons(1,Nil))))` vaut `(Cons(3,Cons(3,Cons(3,Nil))))`

F `f` ne termine sur aucune entrée

G `f (Cons(3,Cons(2,Cons(1,Nil))))` vaut `(Cons(3,Cons(3,Cons(1,Nil))))`

H `(f Nil)` vaut `Nil`

Question 15 ♣ On définit le type `intlist` par `type intlist = Nil | Cons of int*intlist;;`
On définit la fonction `f` par

```
let rec f (l:intlist):int = match l with
|Nil-> 1
|Cons(x,ll)-> x*(f ll)
```

A `f (Cons(3,Cons(2,Cons(1,Nil))))` vaut `(3,2,1)`

B `f (Cons(3,Cons(2,Cons(1,Nil))))` vaut `9`

C `f` ne termine pas pour certaines entrées

D `f (Cons(3,Cons(2,Cons(1,Nil))))` renvoie une erreur

E `f` termine pour toute entrée

F `f (Cons(3,Cons(2,Cons(1,Nil))))` vaut `6`

Question 16 ♣ On définit le type `intlist` par `type intlist = Nil | Cons of int*intlist;;`
On veut implémenter une fonction `f` qui prend en entrée un entier n et renvoie la liste des n premiers entiers dans **l'ordre croissant**. Quelles sont les implémentations correctes parmi les suivantes:

A
```
let rec f (n :int):intlist=
match n with
|0->Cons(n,Nil)
|n-> Cons(n, f (n-1));;
```

B aucune des autres réponses n'est correcte.

C
```
let rec f (n :int):intlist=
let i=n in
if (i=0) then Nil else let (i=i-1) in Cons(i+1,f i);;
```

D
```
let rec f (n :int):intlist=
let i=0 in
if (i=n) then Nil else let (i=i+1) in Cons(i-1,f i);;
```

Question 17 ♣ On veut implémenter une fonction `div` qui prend comme entrées un entier positif a et un entier strictement positif b et renvoie le couple d'entier (q, r) formé du quotient et du reste dans la division euclidienne de a par b, c'est à dire $a = b \times q + r$ avec $0 \leq r < b$. Quelles sont les implémentations correctes parmi les suivantes:

A
```
let div a b = (a/b,a mod b);;
```

B
```
let div a b =
    let q=int_of_float((float_of_int a) /.  (float_of_int b)) in
    (q,a-b*q);;
```

C
```
let rec div a b=
    if a<b then (0,a)
    else
       let (q,r)=div (a-b) b in
       (q+1,r);;
```

Question 18 ♣ Le type `t` défini par `type t = F of int | A of int*int*t ;;`

A est un type produit

B permet de former des séquences de `int` de longueur impaire.

C est un type somme

D est un brave type

E est un type énuméré

F est un type récursif

Question 19 ♣ Le type `t` défini par `type t = int*float` est

A un type somme

B un type synonyme du type `float*int`

C un type synonyme du type `int*float`

D un type récursif

E un type produit

F un type énuméré

Question 20 ♣ On défini `f` par `let f (x:float):float = 3.*.x+.4.;;`

A `f (-.2.)` vaut 0.

B f est une fonction affine

C `f (f 1.)` vaut 10.

D `f (-.2.)` vaut `-.2.`

E `f (f 1.)` vaut 25.

F f est de type `float→float`