ENICarthage ——***——	**DEVOIR SURVEILLÉ**	Novembre 2020 ——***——

SPÉCIFICATION ET VÉRIFICATION FORMELLES

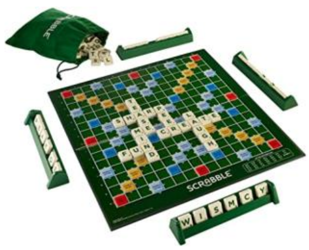
Enseignantes	:	Mme. M. Fourati & Mme. E. Menif	**Nom**	:	
Niveau	:	3ème année ingénierie des systèmes intelligents	**Prénom**	:	
Documents et calculatrices	:	Non autorisées	**Groupe**	:	

SPÉCIFICATION FORMELLE: (10 points)

Exercice 1 : Questions de cours (2 points)

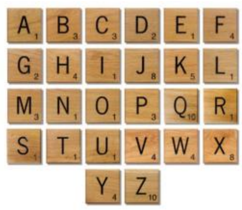
	Vrai	Faux
1. Un langage de programmation peut être un langage de spécification formelle	☐	☐
2. Les besoins énoncés sont le fruit de la phase d'analyse des besoins	☐	☐
3. La spécification est conduite au début de chaque phase de du cycle de vie d'un logiciel	☐	☐
4. La sur-spécification est un élément de la spécification formelle ne correspondant pas à une caractéristique du problème mais plutôt de la solution	☐	☐
5. En présence de la spécification formelle, la spécification en langage naturelle n'est plus nécessaire	☐	☐
6. seq $X \triangleq \{f : \mathbb{N} \rightarrow X\}$	☐	☐
7. L'approche axiomatique fait partie des méthodes orientées modèle	☐	☐
8. Dans Z, tout ensemble est un type	☐	☐

Exercice 2 (8 points)



On s'intéresse à la spécification du jeu de société Scrabble. Ce jeu consiste à former des mots entrecroisés sur une grille avec des lettres de valeurs différentes. Le vainqueur est le joueur qui cumule le plus grand nombre de points à l'issue de la partie. Le jeu se compose, notamment :

- d'un plateau de 225 cases ;
- de 102 jetons dont 2 jokers; un jeton correspond à une lettre et un ombre de points
- de chevalets sur lesquels chaque joueur pose ses lettres.



On ne va spécifier qu'une version simplifiée de ce jeu. On fera abstraction des jokers et des cases multiplicatrices du plateau.

- Le score d'un coup est calculé en additionnant la valeur de toutes les lettres des nouveaux mots formés.
- Pour commencer, chaque joueur va tirer au hasard 7 jetons dans le sac. Le joueur peut avoir plusieurs jetons identiques
- Chaque joueur doit former un mot (suite de jetons) à partir des jetons qu'il détient.
- Après avoir placé un mot, le joueur doit piocher autant de jetons que ceux placés sur le plateau pour qu'il ait toujours un nombre total de 7 jetons sur son chevalet.

On considère les types suivants :
Lettre ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
Points $\triangleq$ {1,2,3,4,8,10}
Jetons : **Lettre** $\rightarrow$ **Points**

Nous supposons avoir les opérations **first**, **second** et **Somme**. L'opération **first** prend un couple et retourne le premier élément du couple. L'opération **second** prend un couple et retourne le deuxième élément du couple. L'opération **Somme** qui prend un ensemble d'entiers et retourne la somme de ces entiers.

1. Définir les ensembles **Voyelles** et **Consonnes** à partir de l'ensemble **Lettre**.

……

………………………………………………………………………………………………………..

2. Définir l'opération **Jouer** qui prend l'ensemble des jetons que possède un joueur, l'ensemble des jetons qu'il va jouer pour former son mot et retourne l'ensemble des jetons qui lui reste.

....
....

3. Définir l'opération **NombreJetons** qui prend l'ensemble des jetons en possession du joueur et retourne le nombre des jetons.

....
....

4. Définir l'opération **PrendreJetons** qui prend l'ensemble des jetons en possession du joueur après avoir joué, l'ensemble des jetons qu'il a pioché et retourne l'ensemble des jetons qui seront sur son chevalet.

....
....

5. En considérant le mot formé par un joueur comme une suite de jetons. Définir l'opération **ScoreMot** qui prend le mot composé par le joueur et retourne le nombre de points correspondant à ce mot.

....
....

6. Définir l'opération **VoyellesMot** qui prend l'ensemble des jetons en possession du joueur et retourne le nombre des voyelles présentes

....
....

<u>VÉRIFICATION FORMELLE</u> : (10 points)

<u>Exercice 1 : Questions de cours</u> (2 points)

- La logique LTL est plus expressive que la logique CTL. ☐ ☐
- La logique CTL est plus expressive que la logique LTL. ☐ ☐
- **F***p* est vraie lorsque *p* est vrai dans l'état courant. ☐ ☐
- *p***AU***q* est vraie lorsque *q* est vrai dans l'état courant. ☐ ☐
- **GF***p* est vrai s'il existe une exécution dans laquelle *p* est toujours vrai et à un certain moment, *p* devient toujours faux. ☐ ☐
- Le model checking consiste à donner une preuve formelle de la satisfaction d'une formule. ☐ ☐

<u>Exercice 2 : Modélisation et produit d'automates</u> (5 points)

Considérons un système (très simplifié) de gestion des prix des produits dans un supermarché. Le système consiste en trois processus : un lecteur de code à barre (LCB) qui lit le code et communique sa référence au programme d'attribution des prix (AP). En recevant cette référence, le programme AP transmet le prix de l'article à l'imprimante (IMP) qui génère un ticket (contenant la référence de l'article et son prix). Uniquement le ticket est important, nous ne nous intéressons pas à son contenu.

Sachez que le but de l'exercice est de synchroniser les trois processus LCB, AP et IMP par envoie/réception de message et qu'il peut y avoir des actions d'envoie/réception et d'autres internes aux processus.

1. Modéliser le processus LCB par une structure de Kripke. Indiquer l'état initial et les actions de ce dernier.

2. Modéliser le processus AP par une structure de Kripke. Indiquer l'état initial et les actions de ce dernier.

3. Modéliser le processus IMP par une structure de Kripke. Indiquer l'état initial et les actions de ce dernier.

4. Synchroniser les trois processus par envoie/réception de messages.

<u>Exercice 2 : Automate temporisé</u> (3 points)

Nous voulons modéliser une sonnerie, lorsque l'utilisateur appuie sur le bouton *push*, la sonnerie va sonner durant exactement 5 secondes. Modéliser la sonnerie par un automate temporisé en évitant d'aboutir à un blocage sur les actions franchissables.

1. Modélisez par un automate temporisé M la sonnerie.

2. Donnez une exécution de l'automate M contenant exactement 4 transitions (5 configurations) et qui revient à l'état initial.

……………………………………………………………………………………………………

……………………………………………………………………………………………………

……………………………………………………………………………………………………

……………………………………………………………………………………………………

Éléments de la syntaxe du langage Z

Les relations

Notation	Sens	Définition
dom R	Domaine	$\{x{:}X \mid (\exists y{:}Y \bullet (x,y) \in R)\}$
ran R	Codomaine	$\{y{:}Y \mid (\exists x{:}X \bullet (x,y) \in R)\}$
id R	Identité	$\{x{:}X \bullet x \mapsto x\}$
$R^{\sim}$	Inverse	$\{y{:}Y, x{:}X \mid (x,y) \in R\}$
$R \fatsemi R'$	Composition	$\{x{:}X, z{:}Z \mid (\exists y{:}Y \bullet (x,y) \in R \wedge (y,z) \in R')\}$
R^k	Composition récurrente	
$R(\!|S|\!)$	Image relationnelle	$\{y{:}Y \mid (\exists x{:}S \bullet (x,y) \in R)\}$
$S \lhd R$	Restriction du domaine	$\{x{:}X, y{:}Y \mid x \in S \wedge (x,y) \in R\}$
$R \rhd S'$	Restriction du codomaine	$\{x{:}X, y{:}Y \mid y \in S' \wedge (x,y) \in R\}$
$S \ntriangleleft R$	Soustraction de domaine	$(X \setminus S) \lhd R$
$R \ntriangleright S'$	Soustraction de codomaine	$R \rhd (Y \setminus S')$
$R \oplus R'$	Surcharge	$\{x{:}X, y{:}Y \mid (x,y) \in R' \vee (x \notin \mathrm{dom}\ R' \wedge (x,y) \in R)\}$

Les séquences

Notation	Sens	Définition
$\langle x_1, x_2, \ldots, x_n \rangle$ ou $[x_1, x_2, \ldots, x_n]$	Notation alternative	$\{1 \mapsto x_1, 2 \mapsto x_2, \ldots, n \mapsto x_n\}$ ou $\{(1,x_1), (2,x_2), \ldots, (n,x_n)\}$
$\langle\rangle$ ou $[]$	Séquence vide	
$s(i)$	$i^{\text{ème}}$ élément si $i \in 1..\#s$	
$\mathrm{seq}_1\ X$	Séquences finies non vides	$\triangleq \{f : \mathrm{seq}\ X \mid \#f > 0\}$
$\#s$	Cardinal	
$s \frown t$	concaténation	$\triangleq s \cup \{n : \mathrm{dom}\ t \bullet n + \#s \mapsto t(n)\}$
rev s	Inversion	$(\lambda n : \mathrm{dom}\ s \bullet s(\#s . n + 1))$
head s	Premier élément	$\forall s{:}\mathrm{seq}_1\ X \bullet \mathrm{head}\ s = s(1)$
tail s	Liste sans le premier élément	$\forall s{:}\mathrm{seq}_1\ X \bullet \mathrm{tail}\ s = (\lambda n : 1..\#s.1 \bullet s(n+1))$
last s	Dernier élément	$\forall s{:}\mathrm{seq}_1\ X \bullet \mathrm{last}\ s = s(\#s)$
front s	Liste sans le dernier élément	$\forall s{:}\mathrm{seq}_1\ X \bullet \mathrm{front}\ s = (1..(\#s.1)) \lhd s$
$s \upharpoonright A$	Filtre une séquence en ne considérant que les éléments de A	squash $(s \rhd A)$
$I \upharpoonleft s$	Extrait une sous-séquence formée d'éléments avec des indices de I	squash $(I \lhd s)$
$\langle A, B \rangle$ partition C	$A \cap B = \emptyset \ \wedge\ A \cup B = C$	$\_\ \mathrm{partition}\ \_ : (\mathbb{N} \pfun \mathbb{P}X) \leftrightarrow \mathbb{P}X$

Les bags

Notation	Sens	Définition
$[\![\,]\!]$	Multi.ensemble vide	
$[\![x_1, x_2, \ldots, x_n]\!]$	Multi.ensemble en extension	$x_1, x_2, \ldots, x_n{:}X$
count b	Nombre de chaque élément x dans b	$\forall b{:}\mathrm{bag}\ X \bullet \mathrm{count}\ b = (\lambda x : X \bullet 0) \oplus b$
$b \# x$	Nombre d'occurrences de x dans b	$\forall x{:}X, b{:}\mathrm{bag}\ X \bullet b \# x = \mathrm{count}\ b\ x$
$n \otimes b$	Mise à l'échelle	$\forall n : \mathbb{N}, x : X, b : \mathrm{bag}\ X \quad (n \otimes b) \# x = n * (b \# x)$
$x \sqsubset b$	Appartenance	$\forall x{:}X, b{:}\mathrm{bag}\ X \bullet (x \sqsubset b \Leftrightarrow x \in \mathrm{dom}\ b)$
$b \sqsubseteq c$	Inclusion	$\forall b, c{:}\mathrm{bag}\ X \bullet b \sqsubseteq c \Leftrightarrow (\forall x{:}X \bullet b \# x \leq c \# x)$
$b \uplus c$	Union	$\forall b, c{:}\mathrm{bag}\ X, x{:}X \bullet (b \uplus c) \# x = b \# x + c \# x$
$b \cup c$	Différence	$\forall b, c{:}\mathrm{bag}\ X, x{:}X \bullet (b \cup c) \# x = \max\{b \# x . c \# x, 0\}$