Feuille de TD 1 de Spécifications Formelles
M1 2012-2013

Les fichiers en format pdf des projections du cours sont disponibles, au fur et à mésure, sur la page :
http ://www.ibisc.univ-evry.fr/~serena

1 Introduction aux *Abstract Machines*

Exercice 1

Reprendre l'opération *serve_next* de l'ébauche de spécification de la machine Tickets de la slide 12. On veut modifier cette opération en *serve_two*, l'idée étant que, sur l'écran, ce sont les 2 prochaines personnes servies, qui apparaîssent. Ecrire alors la spécification complète de cette opération, precondition et body inclus. Vérifier (à la main) que l'invariante de Tickets est conservée (à condition que l'initialisation soit reasonable, bien sûr).

Exercice 2

Ce fois ci, on veut modifier l'ébauche de spécification de la machine Tickets de la slide 14. Supposons que l'invariante contient une condition (un conjoint) de plus : $next \leq serve + 20$, qui indiquera que ne doivent pas exister plus que 20 tickets pas encore utilisés. Pourquoi l'opération *take_ticket* de la slide 14 n'est pas cohérente avec cette nouvelle invariante ? Comment pourrait on modifier la precondition de take-ticket pour rétablir la cohérence avec cette nouvelle invariante ?

Exercice 3

Est-ce que l'operation

$$
\begin{array}{l}
tt \leftarrow \textbf{replace\_ticket} \mathrel{\widehat{=}} \\
\quad \textbf{PRE } true \\
\quad \textbf{THEN } next := next - 1 \\
\quad \textbf{END ;}
\end{array}
$$

est cohérent avec l'invariante de la machine *Ticket* ?

Exercice 4

Réprendre la spécification de Tickets de p. 14 et ajouter une operation *reset*, qui renvoi l'état initial.

Exercice 5

Modifier encore la spécification de Tickets, de façon telle que on peut distribuer seulement des tickets numerés jusqu'à 500. NB : Il faudra modifier l'invariante de la machine, et aussi l'opération *take_tickets*.

Exercice 6

Ajoutez une autre variable *record* à la machine. Cette variable compte le nombre des personnes qui ont pris un ticket depuis la dernière opération **serve_next**. Elle sera incrémentée lorsqu'on invoque **take_ticket**, et resettée à 0 lorsqu'on invoque **serve_next**. Ajoutez aussi une opération de requete, qui montre la valeur de *record* à chaque moment.

Exercice 7

Est-ce que l'opération

$$
\begin{array}{l}
\textbf{undo\_serve} \mathrel{\widehat{=}} \\
\quad \textbf{PRE}\ \mathit{true} \\
\quad \textbf{THEN}\ \mathit{serve} := \mathit{serve} - 1 \\
\quad \textbf{END}\,;
\end{array}
$$

est cohérent avec l'invariante de la machine *Ticket* ?

Exercice 8

Un parking a 640 places. Décrivez une *abstract machine* qui spécifie un système pour controler les voitures qui entrent dans le parking. Vous devez tenir le compte de nombre des voitures dans le parking à chaque moment, et fournir trois opérations :

- **enter**, qui régistre l'entrée d'une voiture ; ça se passe seulement si le parking n'est pas plein.
- **leave**, qui régistre la sortie d'une voiture du parking.
- $nn \leftarrow$ **query**, qui montre le nombre des voitures dans le parking à chaque fois.

2 Rappels de logique, et substitutions

Exercice 1

1. Soit la formule $\forall x \ \exists y x = 2 * y$, interprété de la façon "naturelle" sur $\mathbb{N}$. Quelle est sa valeur ?
2. Même formule qu'avant, mais sans le quantificateurs existentiel :

   $$\exists y y = 2 * x$$

   et toujours la même inteprétation. Cette fois ci, x est libre, donc l'évaluation de la formule donne lieu à un ensemble. Lequel ?
3. Avec la formule $m \leq x \wedge x \leq n$, que l'on interprété toujours sur $\mathbb{N}$ de la façon naturelle, on a 3 variables libres, ainsi c'est un ensemble de triplets que l'on est en train de définir. Lequel ?
4. Est il toujours vrai que, peut importe l'intérprétation, la formule

   $$\exists y \forall x r(x, y) \Rightarrow \forall x \exists y r(x, y)$$

   est toujours vraie ? Justifiez la réponse.
5. Dire quel est le résultat de la substitution suivante :

   $$\forall n(n \in \mathbb{N} \Rightarrow (serve < 4 + n^2 \vee n < next))[n/next]$$

3 Espaces d'états, et preconditions

Exercice 1

Soit C un ensemble qui contient une seule variable, $office$ dont le type est déclaré par la formule ensembliste : $office \subseteq \{Mike, Nell, Olivia\}$.

1. Ecrire l'espace d'états E correspondant à C.
2. Soit S l'instruction $office := office \cup Olivia$. Comme vu en cours pour l'exemple des slide 27-28, indiquer graphiquement comme chaque état de E change une fois effectuée S.
3. En utilisant le diagramme que vous venez d'écrire, calculer ce sous-ensemble des états de E qui <u>assurent</u> que la condition $P = card(office) = 2$ vaut. En autre mots, calculez (à la main) exactement le états de E où est vraie la *weakest precondition*

   $$\underbrace{[office := office \cup \{Olivia\}]}_{S} \underbrace{(card(office) = 2}_{P}$$

4 Examples des solutions

4.1 Introduction aux *Abstract Machines*

Exercice 1

Une solution possible est de produire les deux numéros qu'on est en train de servir.

$$
\begin{array}{l}
ss1, ss2 \leftarrow \textbf{serve\_two} \mathrel{\widehat{=}} \\
\quad \textbf{PRE}\ serve \leq next + 2 \\
\quad \textbf{THEN}\ ss1, ss2, serve := serve + 1, serve + 2, serve + 2 \\
\quad \textbf{END}\ ;
\end{array}
$$

Exercice 2

La precondition de *take_ticket* actuelle est *True* (très faible !). Or si on appele cette opération dans un état où $next = serve + 20$, on violera la nouvelle invariante.

Pour empêcher cette incohérence, il faudrait avoir $next < serve + 20$ dans la precondition de *take_ticket*.

Exercice 3

L'opération **replace_ticket** n'est pas cohérente avec l'invariante de la machine *Ticket*, car si elle est invoquée lorsque $serve = next$, alors on obtient un état dans lequel $serve > next$, qui falsifie l'invariante.

Exercice 4

L'opération **reset** n'a aucun input ni output ; la précondition est *true* est le résultat est de initialiser les variables *serve* et *next* à 0.

$$
\begin{array}{l}
\leftarrow \textbf{reset} \mathrel{\widehat{=}} \\
\quad \textbf{PRE}\ true \\
\quad \textbf{THEN}\ serve, next := 0, 0 \\
\quad \textbf{END}\ ;
\end{array}
$$

Exercice 5

Une solution possible est

MACHINE *Ticket*
VARIABLES $serve, next$
INTIALISATION $serve, next := 0, 0$
INVARIANT $serve \in \mathbb{N} \wedge next \in \mathbb{N} \wedge serve \leq next \wedge serve \leq 500$

OPERATIONS
$ss \leftarrow$ **serve_next** $\widehat{=}$
PRE $serve < next$
THEN $ss, serve := serve + 1, serve + 1$
END ;
$tt \leftarrow$ **take_ticket** $\widehat{=}$
PRE $next < 500$
THEN $tt, next := next, next + 1$
END
END

Exercice 6

Une solution possible est

MACHINE *Ticket*
VARIABLES $serve, next, record$
INTIALISATION $serve, next, record := 0, 0, 0$
INVARIANT $serve \in \mathbb{N} \wedge next \in \mathbb{N} \wedge next \in \mathbb{N} \wedge serve \leq next$
OPERATIONS
$ss \leftarrow$ **serve_next** $\widehat{=}$
PRE $serve < next$
THEN $ss, serve := serve + 1, serve + 1 \wedge record := 0$
END ;
$tt \leftarrow$ **take_ticket** $\widehat{=}$
PRE $next < 500$
THEN $tt, next := next, next + 1 \wedge record := record + 1$
END
$nn \leftarrow$ **query** $\widehat{=}$
PRE $true$
THEN $nn := record$
END
END

Exercice 7

No, pourquoi ?

Exercice 8

Une solution possible est

MACHINE *Parking*
VARIABLES $n\_voiture$
INTIALISATION $n\_voiture := 0$
INVARIANT $n\_voiture \in \mathbb{N} \wedge n\_voiture \leq 640$
OPERATIONS
enter $\widehat{=}$
  PRE $n\_voiture < 640$
  THEN $n\_voiture := n\_voiture + 1$
  END ;
leave $\widehat{=}$
  PRE $n\_voiture > 0$
  THEN $n\_voiture := n\_voiture - 1$
  END
$nn \leftarrow$ **query** $\widehat{=}$
  PRE *true*
  THEN $nn := n\_voiture$
  END
END

4.2 Rappels de logique, et substitutions

Exercice 1

1. La formule $\forall x\ \exists y x = 2 * y$ est fausse. Considèrez le cas pour $x = 5$.
2. L'ensemble des nombres pairs.
3. L'ensemble des triplettes (a, b, c) tels que $a \leq b \leq c$.
4. S'il existe un y tel que pour tous les x, $r(x, y)$, alors pour tous les x, la même y est tel que $r(x, y)$.
5. le résultat est

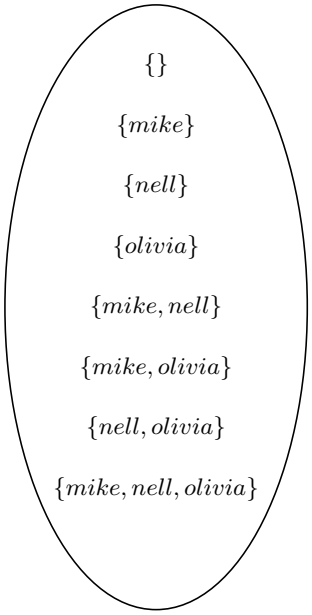
$$\forall m(m \in \mathbb{N} \Rightarrow (serve < 4 + m^2 \vee m < n))$$

   Notez qu'il faut remplacer les occurrences lié de n avec m.

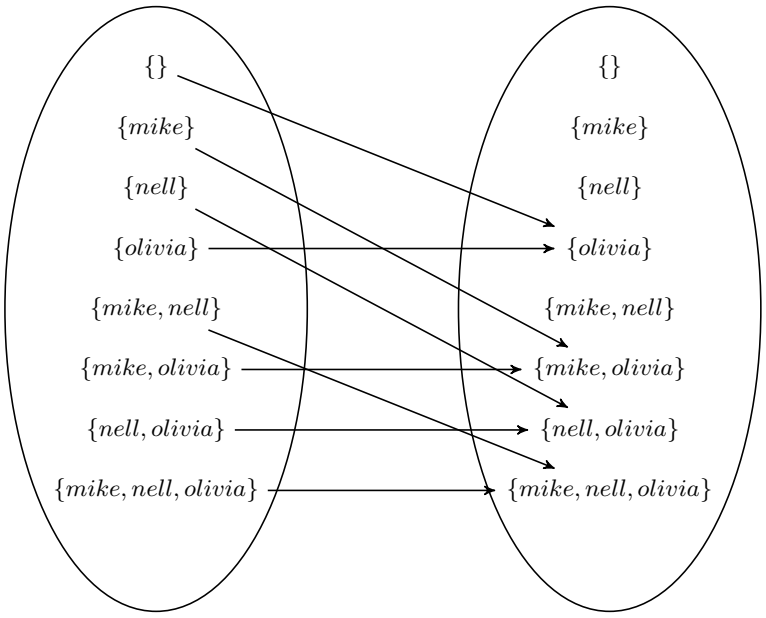
4.3 Espaces d'états, et preconditions

Exercice 1

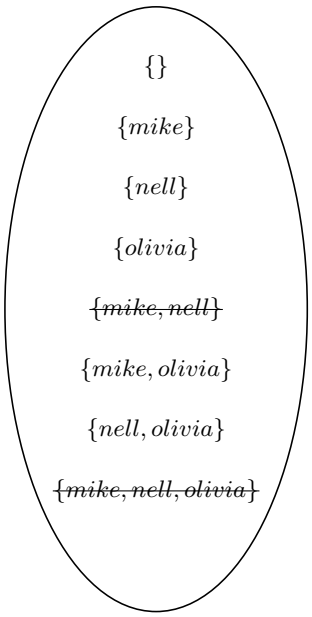
–



–



–


{}
{mike}
{nell}
{olivia}
{mike, nell}
{mike, olivia}
{nell, olivia}
{mike, nell, olivia}