Correction TD 1 de Model Checking

Modélisation des systèmes réactifs

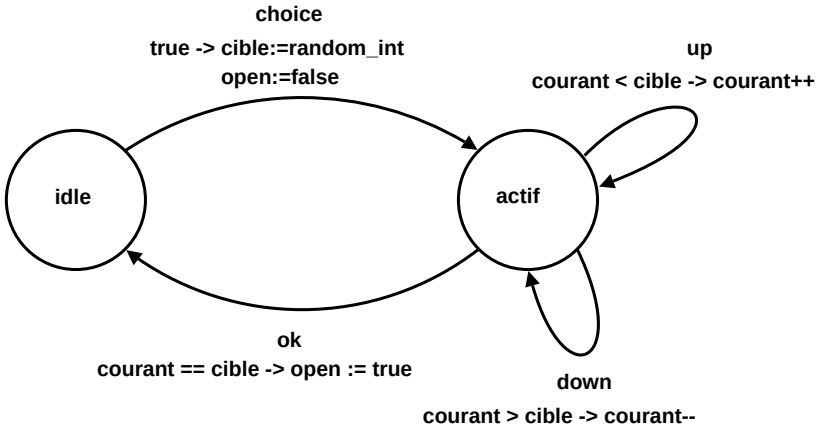
Exercice 1 (Exemple de l'ascenceur.)**.** *Le système de contrôle d'un ascenceur (pour 3 étages) est défini par :*

- *le contrôleur garde en mémoire l'étage courant et l'étage cible.*
- *en mode actif, quand l'étage cible est atteint, les portes s'ouvrent et le contrôleur passe en mode attente.*
- *en mode actif, quand l'étage cible est plus élevé que l'étage courant, le contrôleur fait s'élever l'ascenceur.*
- *en mode actif, quand l'étage cible est moins élevé que l'étage courant, le contrôleur fait descendre l'ascenceur.*
- *en mode attente, il se peut que quelqu'un entre dans l'ascenceur et choisisse un nouvel étage cible. L'ascenceur ferme alors les portes et redevient actif.*
- *initialement, l'ascenceur est à l'étage 0 et en mode attente.*

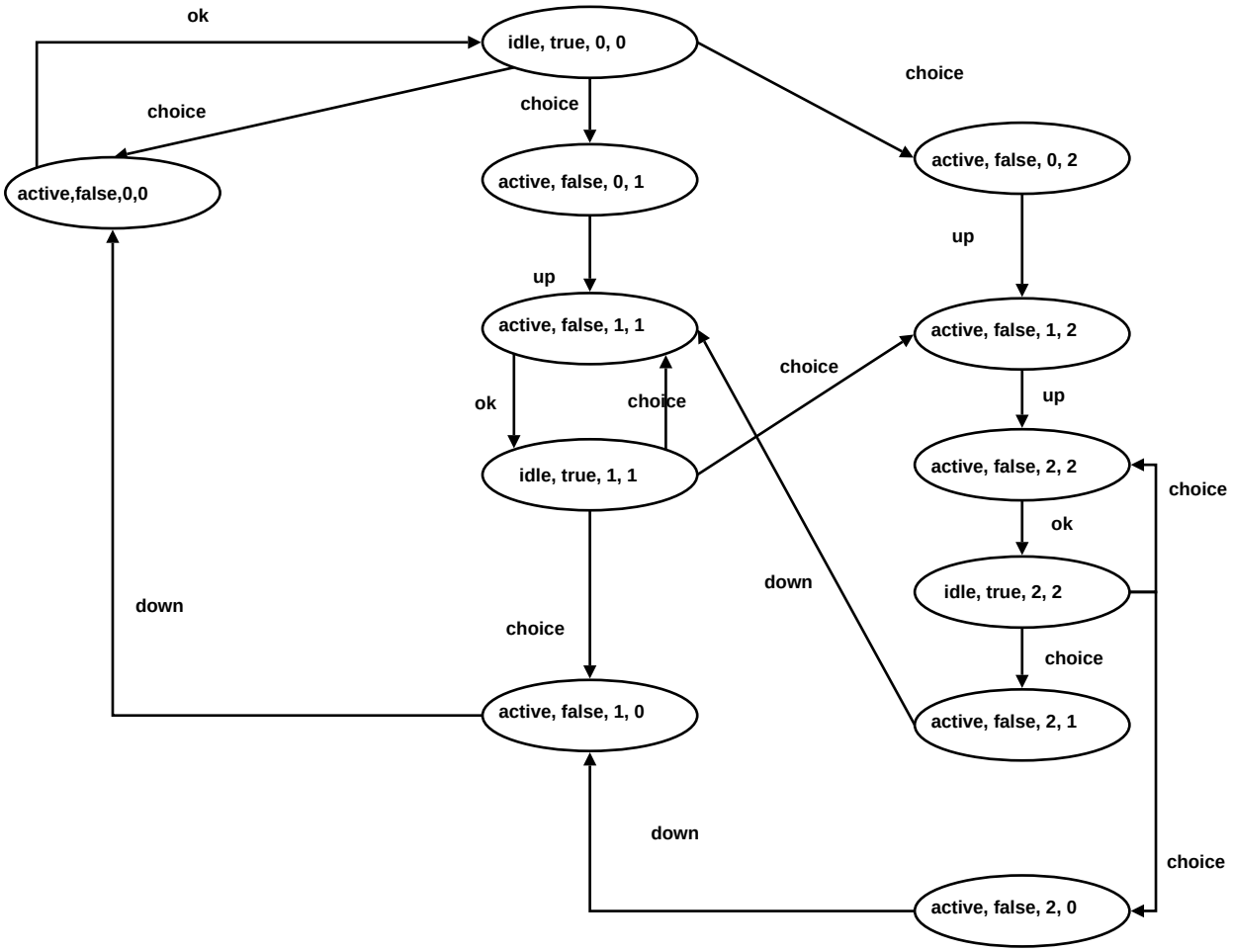
Questions : 1. Proposez une machine à états modélisant le contrôle de l'ascenceur (définition formelle et dessin). 2. Définissez et dessinez le système de transitions correspondant (en vous limitant aux configurations accessibles depuis l'état initial). 3. Est-ce que les portes peuvent s'ouvrir quand l'ascenceur est actif ?

Correction.

1. Voici ma machine à états. Les variables sont `courant :int[0..2]`, `cible :int[0..2]` et `open :bool`. L'action `random_int` retourne un entier entre 0 et 2 de manière non déterministe.



2. L'état initial est : état de contrôle `idle`, `(open, cible, courant) := (true, 0, 0)`.



3. Non, trivial ici vu la modélisation.

□

Exercice 2. *Soit un système de transitions $S = \langle Q, T, \rightarrow \rangle$ et $q_0 \in Q$ une configuration initiale. Montrez que :*

1. *l'ensemble d'accessibilité post$^*(q_0)$ est le plus petit invariant de S contenant q_0 ;*
2. *il existe $k \in \mathbb{N}$ tel que post$^*(q_0) = \bigcup_0^k post^i(q_0)$.*

Correction.

1. On procède en deux phases. D'abord on montre que post$^*(q_0)$ est un invariant de S contenant q_0, puis on montre que c'est le plus petit. (a) Par définition, post$^*(q_0)$ contient q_0. Montrons que post$^*(q_0)$ est un invariant de S, i.e. montrons que post(post$^*(q_0)$) $\subseteq$ post$^*(q_0)$. Soit $x \in$ post$^*(q_0)$ et $x' \in$ post(x). Par définition, il existe t' tq $x \xrightarrow{t'} x'$. Par définition, on sait aussi qu'il existe $i \geq 0$ tel que $x \in$ post$^i(q_0)$. On en déduit qu'il existe des configurations $x_1, \ldots, x_{i-1}$ et $t_1, \ldots, t_i \in T$ tels que $q_0 \xrightarrow{t_1} x_1 \xrightarrow{t_2} \ldots \xrightarrow{t_{i_1}} x_{i_1} \xrightarrow{t_i} x$. On peut ainsi écrire que : $q_0 \xrightarrow{t_1} x_1 \xrightarrow{t_2} \ldots \xrightarrow{t_{i_1}} x_{i_1} \xrightarrow{t_i} x \xrightarrow{t'} x'$. On déduit ainsi que $x' \in$ post$^{i+1}(q_0)$, et donc $x' \in$ post$^*(q_0)$ Ceci montre que post$^*(q_0)$ est un invariant de S. On a aussi montré qu'il contenait q_0 juste avant. (b) Montrons que post$^*(q_0)$ est le plus petit invariant de S à contenir q_0. Soit I un invariant de S contenant q_0. On montre par récurrence que pour tout i, I contient post$^i(q_0)$. La propriété est vraie pour $i = 0$ par définition de I. Si on suppose la propriété vraie au rang i, alors post$^i(q_0) \subseteq I$. Or post$^{i+1}(q_0) =$ post(post$^i(q_0)$), donc post$^{i+1}(q_0) \subseteq$ post(I) (croissance de post et hypothèse de récurrence) et post$^{i+1}(q_0) \subseteq I$ (I est un invariant). Ceci montre par récurrence que pour tout i, I contient post$^i(q_0)$. Or post$^*(q_0) = \bigcup_0^\infty$ post$^i(q_0)$. D'où la conclusion.

2. Montrons qu'il existe $k \in \mathbb{N}$ tel que $\text{post}^*(q_0) = \bigcup_0^k \text{post}^i(q_0)$. On note $C_n = \bigcup_0^n \text{post}^i(q_0)$. On remarque d'abord que par définition, pour tout n on a $C_n \subseteq C_{n+1}$ (croissance). On va ensuite montrer que s'il existe q tq $C_q = C_{q+1}$, alors pour tout $n \geq q$, $C_n = C_q$ et $\text{post}^*(q_0) = C_q$ (stabilisation). Soit q tq $C_q = C_{q+1}$. On veut montrer que pour tout $n \geq q$, $C_n = C_q$. On procède par récurrence sur $i + q$ (en faisant varier i). Le cas $i = 1$ est vraie par hypothèse sur q. On suppose maintenant que $C_q = C_{q+i}$ et on veut montrer que $C_q = C_{q+i+1}$. La différence entre C_q et C_{q+i+1} est (hyp. de récurrence) l'ensemble $\text{post}^{q+i+1}(q_0)$. Or $\text{post}^{q+i+1}(q_0) = \text{post}(\text{post}^{q+i}(q_0))$. Or $\text{post}^{q+i}(q_0) \subseteq C_{q+i}$ donc $\text{post}^{q+i}(q_0) \subseteq C_q$ (hyp. de récurrence). On en déduit que $\text{post}(\text{post}^{q+i}(q_0)) \subseteq \text{post}(C_q) \subseteq C_{q+1}$ (croissance de post et définition de C_n) et enfin que $\text{post}^{q+i+1}(q_0)) \subseteq C_q$ (cas de base de la récurrence). Donc on a montré par récurrence que s'il existe q tq $C_q = C_{q+1}$, alors pour tout $n \geq q$, $C_n = C_q$. Ainsi pour tout $i < q$, $C_i \subseteq C_q$ et pour tout $i > q$, $C_i = C_q$. On en déduit que $\text{post}^*(q_0) = C_q$. Voici qui démontre le "lemme" de stabilisation. On prouve maintenant le résultat final. Soit la suite infinie des C_n. C'est une suite croissante (lemme de croissance), et chaque C_i est inclus dans Q l'ensemble des configurations du système S. Or, par hypothèse sur S, Q est fini (c'est le cadre du cours). On en déduit qu'il existe nécessairement k tq $C_k = C_{k+1}$, sinon si toutes les inclusions étaient strictes la suite $(|C_n|)$ tendrait vers l'infini et dépasserait $|Q|$. On utilise le lemme de stabilisation pour conclure.

□

Exercice 3 (Co-accessibilité)**.** *Nous avons vu comment vérifier l'invariance et l'accessibilité à partir du calcul des états accessibles ("calcul en avant"). On peut aussi vérifier ces propriétés en calculant "en arrière". On définit informellement $pre(q)$ comme les configurations à partir desquelles on peut atteindre q_0.*

- *définir formellement la relation pre (s'inspirer de post)*
- *On appelle ensemble de co-accessibilité de q l'ensemble $pre^*(q)$. Que représente-t-il intuitivement ?*
- *Donner un algorithme pour calculer $pre^*(q)$. Justifiez la terminaison.*
- *Comment vérifier l'invariance et l'accessibilité à partir de la co-accessibilité ?*
- *Montrer qu'il existe $k \in \mathbb{N}$ tel que $pre^*(q) = \bigcup_0^k pre^i(q)$.*

Correction.

1. On peut définir pre par $\text{pre} = \{(q', q) \in Q \times Q | \exists t \in T, q \xrightarrow{t} q'\}$. On peut aussi dire que $\text{pre} = \text{post}^{-1}$ en utilisant la notion de relation inverse ($(q', q) \in \text{pre}$ ssi $(q, q') \in \text{post}$).

2. L'ensemble de co-accessibilité de q représente toutes les configurations du système à partir desquelles on peut atteindre q.

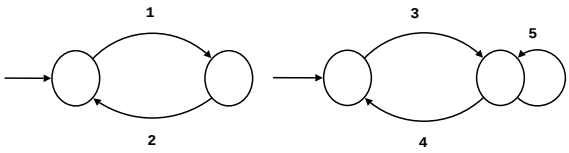
3. Adaptez directement l'algorithme pour post, mais en calculant les transitions $\xrightarrow{t}$ "en arrière" (attention : cela donne en général un ensemble fini de configurations plutôt qu'une seule configuration).

4. Soit un système S avec un état initial q_0, une propriété d'invariance I et une propriété d'accessibilité A (I et A sont des ensembles de configurations). Alors : A est atteignable ssi $q_0 \in \text{pre}^*(A)$; I est vérifié ssi $q_0 \notin \text{pre}^*(Q \backslash I)$.

5. S'inspirer directement de la démonstration précédente.

□

Exercice 4 (Concurrence)**.** *Soit les machines concurrentes suivantes.*



Quelles sont les transitions du système concurrent dans les cas suivants :

1. *sémantique synchrone ;*
2. *sémantique asynchrone (1) ;*
3. *sémantique asynchrone (2) ;*
4. *sémantique synchrone + synchronisation entre 1 et 3 ;*
5. *sémantique asynchrone (2) + synchronisation entre 1 et 3 ;*

Quel est le lien entre vecteur de synchronisations et rendez-vous ?

Correction.

1. les transitions de la machine produit sont : 1x3, 1x4, 1x5, 2x3, 2x4, 2x5.
2. les transitions de la machine produit sont : 1xε, 2xε, εx3, εx4, εx5.
3. les transitions sont celles de la question 1 plus celles de la question 2.
4. les transitions de la machine produit sont : 1x3, 2x4, 2x5.
5. les transitions de la machine produit sont : 1x3, 2x4, 2x5, 2xε, εx4, εx5.
6. Un RDV est un vecteur de synchronisation ne contenant que deux transitions.

□

Exercice 5 (Sûreté)**.** *On se donne un système de transitions S dont certaines transitions sont distinguées et correspondent à des opérations d'aquisition de verrou (`lock`), de rendu de verrou (`unlock`), de lecture (`read`) et d'écriture (`write`). On se donne la propriété φ suivante :*

Si on ne regarde que les `lock` et `unlock` : `unlock` est toujours précédé directement de `lock` ET une suite arbitraire de `read`,`write` est toujours précédée directement d'un `lock` .

Questions :

1. *Est-ce que φ est une propriété de sûreté ? Sinon modifiez là en conséquence.*
2. *Écrivez un automate observeur pour vérifer φ (ou sa modification) et expliquez la nouvelle propriété à vérifier.*
3. *On considère maintenant la spécification :*

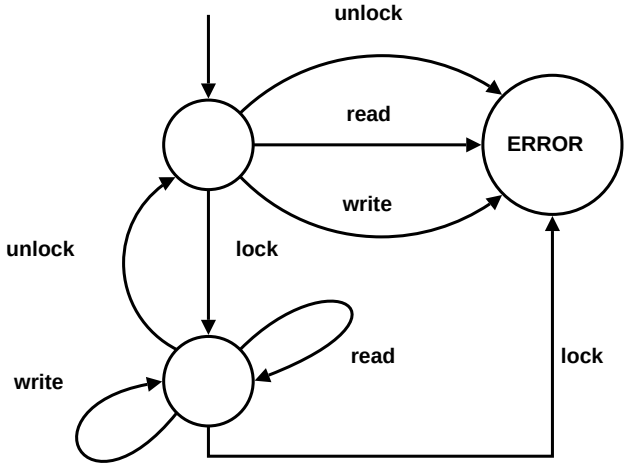
   Si on ne regarde que les `lock` et `unlock` : `unlock` est toujours précédé directement de `lock` `lock` est toujours suivi directement de `unlock`, ET une suite arbitraire de `read`,`write` est toujours précédée directement d'un `lock` et fermée directement par un `unlock`.

   Est-ce toujours une propriété de sûreté ?

Correction.

1. φ est bien une propriété de sûreté car (intuitivement) on peut reconnaitre une exécution incorrecte en considérant juste une portion finie du passé de cette exécution. Ici, il suffit de regarder la trace d'exécution pour vérifier que les `lock, unlock, read, write` sont bien mis correctement.

2. Voilà l'automate observeur ci-dessous. La propriété à vérifier (dans le système produit) est que l'état de contrôle ERROR de l'automate oberveur n'est pas atteignable.



3. Cette propriété n'est plus de la sûreté : pour savoir que tout `lock` est suivi d'une séquence éventuelle de `read`/`write` puis d'un `unlock`, il faut "regarder" le futur de l'exécution. Ainsi, une exécution (infinie) fautive commence par exemple par un `lock` et n'a jamais de `unlock`. Regarder le passé de cette exécution ne permet jamais de dire qu'elle est fautive.

□