

Automates et Vérification Formelle

Séance d'évaluation (2 heures)

Généralités

L'outil à utiliser est le model-checker UPPAAL. Le rendu attendu sont les fichiers

- *lift_stating_point.xml* (Partie I)
- *lift.q* (Partie II)
- *lift_strategy.xml* (Partie III)
- *lift_extended.xml* (Partie IV)

Le tout à envoyer **par mail** avec pour sujet “**UPPAAL Lift**” avant la fin de la séance.

Les réponses textuelles, le cas échéant, sont à inclure dans **le corps du mail**.

Le barème est indiqué à titre indicatif et peut être amené à changer.

Introduction

Le but est de modéliser le contrôle d'un ascenseur composé d'une cage et d'une porte. Des boutons permettent d'appeler et/ou de diriger l'ascenseur vers un étage donné. L'immeuble comporte 5 étages, numérotés de 0 à 4.

Lorsqu'un bouton est *appuyé*, il le reste jusqu'à ce que la porte se ferme à l'étage correspondant.

Le modèle doit permettre de différencier si l'ascenseur est en mouvement ou non.

Partie I: Modèle de départ (5 pts)

Merci de respecter exactement le nommage des variables et patrons.

Déclarations globales:

```
chan openDoor, closeDoor, engineOn, engineOff, goUp, goDown;
```

```
bool btn[5] = {false, false, false, false, false};
```

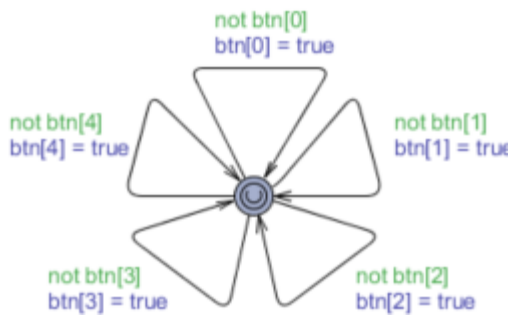
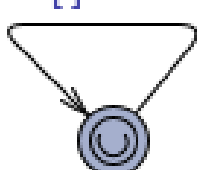
```
bool doorsOpen = false;
```

```
bool engineIsOn = false;
```

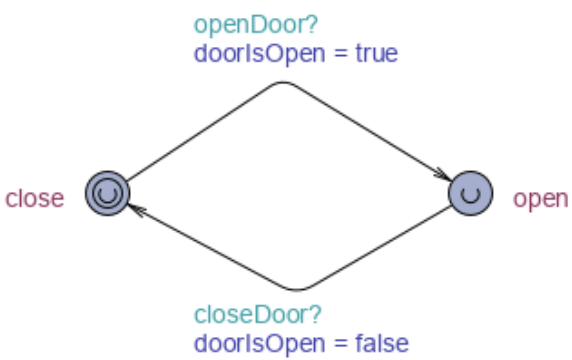
```
int currentFloor = 0;
```

Les canaux de synchronisation permettent de *connecter* le contrôleur à l'ascenseur et à sa porte. Le tableau de booléens *btn* permet de consulter l'état des différents boutons. Les booléens *doorsOpen* et *engineIsOn* permettent de consulter l'état de la porte et de déterminer si l'ascenseur est en mouvement respectivement. L'entier *currentFloor* permet de consulter l'étage où se trouve l'ascenseur.

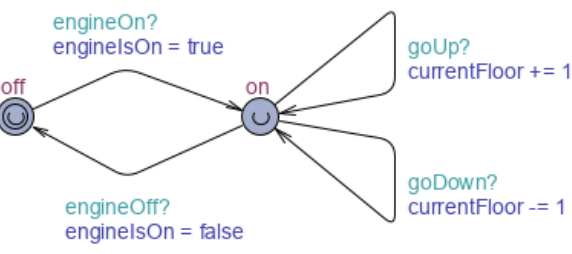
Patron: Env

	<p>$i : \text{int}[0,4]$ not btn[i] btn[i] = true</p> 
<p>L'environnement du système est responsable d'appuyer sur les boutons de manière non déterministe. Les deux automates proposés sont strictement équivalents, utilisez celui que vous préférez. Le champ <i>select</i> pour une transition permet, dans le deuxième cas, de choisir de manière non déterministe un <i>i</i> entre 0 et 4 (inclus).</p>	

Patron: Door

	<p>Ce patron est responsable de se synchroniser en réponse aux canaux openDoor et closeDoor. Il maintient à jour le booléen doorsOpen. Il est aussi responsable de relâcher le bouton correspondant à l'étage courant lors de sa fermeture.</p> <p>a) Le patron présenté sur l'image est incomplet, il manque une action sur l'une des transitions.</p> <p>b) Compléter l'automate pour qu'il soit toujours possible de se synchroniser sur ses canaux.</p>
---	--

Patron: Lift

	<p>Ce patron est responsable de se synchroniser en réponse aux canaux engineOn, engineOff, goUp et goDown. Il maintient à jour le booléen engineIsOn et l'entier currentFloor. Il est aussi responsable d'assurer qu'il ne soit pas possible d'aller plus haut que l'étage 4 et plus bas que l'étage 0.</p> <p>a) Le patron présenté sur l'image est incomplet, il manque les gardes pour répondre à ce dernier point.</p> <p>b) Compléter l'automate pour qu'il soit toujours possible de se synchroniser sur ses canaux.</p>
---	--

Patron: Controler

Ce patron constitue le cœur du système. Il est responsable de contrôler l'ascenseur et sa porte en se synchronisant en émissions sur les différents canaux. **Produisez une première version de ce patron pouvant se synchroniser en émission (c!) de manière entièrement non déterministe.**

System declarations

```
lift = Lift(); env = Env(); door = Door(); controler = Controler();
```

```
system lift, env, door, controler;
```

Encore une fois, merci de respecter le nommage proposé. Attention aux minuscules et majuscules.

Rendus

Une fois les différents patrons complétés, assurez-vous que ça tourne en simulation et **sauvegardez votre modèle** sous le nom *lift_starting_point.xml* (fichier à joindre dans le mail en fin de séance).

Pour éviter d'écraser ce fichier par la suite, sauvegardez à nouveau votre modèle sous un nom différent.

Partie II: Propriétés à vérifier (5 pts)

- P1: Pas de deadlock.
- P2: Il existe un chemin menant à une configuration où tous les boutons sont appuyés.

Exprimez ces propriétés, incluez leurs noms (P1, P2) dans le champ de commentaire, et vérifiez que le modèle de départ respecte ces deux propriétés.

- P3: La porte ne doit jamais être ouverte lorsque l'ascenseur est en mouvement.
- P4: Pour tous les étages, si la porte est ouverte le bouton est appuyé.
- P5: Pour tous les étages, si le bouton correspondant est appuyé, l'ascenseur s'arrête éventuellement à cet étage et la porte s'ouvre.
- P6: Il existe un chemin où l'ascenseur n'atteint jamais l'étage 4.

Idem, mais cette fois-ci le modèle de départ ne permet pas de vérifier ces propriétés. Remarque, certaines de ces propriétés peuvent nécessiter plusieurs formules (une par étage).

Sauvegardez vos requêtes sous le nom *lift.q* (fichier à joindre dans le mail en fin de séance).

Partie III: Stratégie de contrôle (7 pts)

Modifiez votre contrôleur pour répondre à l'ensemble des propriétés. N'hésitez pas à rajouter des déclarations locales à votre contrôleur si besoin. Plusieurs approches sont possibles.

Il est possible d'assurer P1, P2, P3, P4 et P6 relativement simplement (ajouter des gardes suffit). Cependant, cette approche ne permet pas d'assurer que l'ascenseur passe forcément par un étage donné et ne suffit donc pas pour assurer P5. Pour répondre entièrement au problème posé, n'hésitez pas à introduire un nombre conséquent d'états et transitions pour votre contrôleur. A titre d'exemple, la stratégie proposée comme solution finale définit un contrôleur avec 7 états et 12 transitions.

Sauvegardez votre modèle sous le nom *lift_strategy.xml* (fichier à joindre dans le mail en fin de séance). Si certaines propriétés ne sont pas vérifiées, **expliquez pourquoi** dans le corps du mail. **Commentez votre stratégie** dans le corps du mail.

Partie IV: Extensions (3 pts)

Avant d'étendre votre modèle, n'oubliez pas de le sauvegarder sous un nouveau nom pour ne pas écraser le précédent.

- A) Proposez une ou plusieurs nouvelles propriétés (à commenter et sauvegarder dans *lift.q.xml*).
- B) Proposez une nouvelle fonctionnalité, et commentez brièvement vos ajouts dans le corps du mail.

Sauvegardez votre modèle sous le nom *lift_extended.xml*.