Lionel DEBROUX
Savaş Ali TOKMEN
Master 2 Génie Informatique
Option Systèmes et Logiciels Embarqués
UFR IMA, Université Joseph Fourier, Grenoble

Exercice 7: porte de tramway avec passerelle

Cahier de charges

Fonctions

L'application est un programme qui contrôle l'ouverture et la fermeture automatique d'une porte de tramway avec passerelle. Ce contrôleur a quatre éléments dans l'environnement:

- **La porte**: le contrôleur lui envoie les commandes `ouvrir_porte` et `fermer_porte`. L'état de la porte est connu en permanence grâce au signal `porte_ouverte`. On suppose que la porte et son capteur d'ouverture fonctionnent parfaitement bien: la porte ne s'ouvre et se ferme que sur la commande du contrôleur.
- **La passerelle**: sous la porte, le tramway dispose d'une passerelle extractible (utile pour les poussettes, les fauteuils roulants, etc.). Cette passerelle est contrôlée avec les commandes `sortir_pass` et `rentrer_pass`, son état est connu par le signal `pass_sortie`. Comme la porte, la passerelle et son capteur fonctionnent parfaitement bien.
- **L'usager** qui peut appeler `demande_porte` pour demander la porte ou `demande_pass` pour demander la passerelle. Cette demande peut se faire quand le tramway n'est pas en station (où une lumière “porte demandée” ou “passerelle demandée” est allumée) ou quand le tramway est en station (dans quel cas la passerelle sort et la porte s'ouvre, si bien sûr le tramway n'a pas signalé “attention au départ”).
- **Le tramway** fournit au contrôleur deux informations: `en_station` pour signaler que le tramway est en station et `attention_depart` pour signaler que le tramway va partir bientôt. Le tramway attendra pour le signal `porte_pass_ok` (qui signifie que la porte est fermée et la passerelle est rentrée) pour redémarrer.

Critères de sécurité

Il est important que les critères de sécurité suivants soient satisfaits (en supposant que la porte, la passerelle et le tramway fonctionnent correctement):

- La porte ne doit jamais être ouverte quand le tramway n'est pas en station.
- La passerelle ne doit jamais être sortie quand le tramway n'est pas en station.
- La passerelle ne doit jamais bouger (ni rentrer, ni sortir) tant que la porte est ouverte.

Critères d'usabilité

Il y a deux moyens très simples de satisfaire les critères de sécurité:

1. Ne jamais ouvrir la porte ni sortir la passerelle.
2. Ne jamais émettre le signal **porte_pass_ok** (donc le tramway ne part jamais).

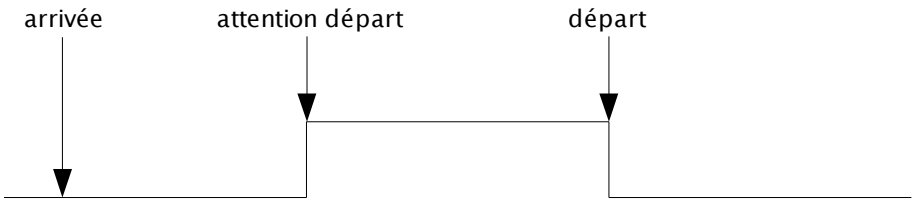
Il est donc important que les critères d'usabilité suivantes soient satisfaits:

- Quand un passager demande la passerelle ou la porte alors que le tramway est en mouvement, cette demande doit être satisfaite (la passerelle sortie si nécessaire puis la porte ouverte) au prochain arrêt.
- Quand un passager demande la passerelle ou la porte alors que le tramway est en station et n'a pas encore signalé **attention_depart** pour cette station et que les portes sont fermées, cette demande doit être satisfaite (la passerelle sortie si nécessaire puis la porte ouverte).
- Quand le tramway signale **attention_depart**, le contrôleur doit fermer la porte, si nécessaire rentrer la passerelle et une fois ces deux actions terminés signaler **porte_pass_ok**.

Implémentation

Signaux internes

Nous définissons un signal interne nommé **vapartir**, qui décrit si le tramway veut partir (est en station et a signalé **attention_depart**). Il agit donc de la façon suivante:



Ce qui donne comme code Lustre:

```
vapartir = en_station and ( attention_depart or prev );
```

Pour des raisons de commodité, nous avons aussi définis les “pre” des signaux **porte_demandee** (**pred**), **pass_demandee** (**prep**), **ouvrir_porte** (**preo**), **sortir_pass** (**pres**) et **vapartir** (**prev**).

Signaux de sortie

porte_demandee

La porte:

- Reste demandée tant que le tramway n'a pas atteint son arrêt
- Devient demandée si la porte (ou la passerelle) est demandée alors que l'arrêt n'a pas été atteint.

De plus, tant que la passerelle est en train de sortir, on laisse le signal **porte_demandee** à vrai pour que les passagers sachent que la porte va ouvrir bientôt.

Cette spécification donne comme code Lustre:

```
porte_demandee = ( sortir_pass or not en_station ) and
                 ( pred or demande_porte or pass_demandee or sortir_pass );
```

pass_demandee

La passerelle:

- Reste demandée tant que le tramway n'a pas atteint son arrêt

- Devient demandée si la passerelle est demandée alors que l'arrêt n'a pas été atteinte.

Cette spécification donne comme code Lustre:

```
pass_demandee = ( not en_station ) and ( prep or demande_pass );
```

ouvrir_porte

La porte:

- Ne doit s'ouvrir que si le tramway est à un arrêt.
- Avant de s'ouvrir, elle doit attendre que la passerelle sorte si ceci est nécessaire (donc la passerelle était demandée).
- Ne s'ouvre que si on l'avait demandé quand le tramway était en mouvement ou que l'on le demande avant que le tramway ait envie de partir.

Cette spécification donne comme code Lustre:

```
ouvrir_porte =   en_station and ( not porte_ouverte ) and ( not vapartir )
                 and not( pass_demandee or sortir_pass )
                 and ( pred or preo or demande_porte );
```

sortir_pass

La passerelle:

- Ne doit sortir que si le tramway est à un arrêt.
- Ne doit pas sortir si la porte est ouverte ou est en train de s'ouvrir.
- Ne sort que si on l'avait demandé quand le tramway était en mouvement ou que l'on le demande avant que le tramway ait envie de partir.

Cette spécification donne comme code Lustre:

```
sortir_pass =   en_station and ( not pass_sortie ) and ( not vapartir )
                and not( preo or porte_ouverte )
                and ( prep or pres or demande_pass );
```

fermer_porte

La porte doit se fermer si le tramway a envie de partir et si elle était ouverte avant.

Cette spécification donne comme code Lustre:

```
fermer_porte = porte_ouverte and vapartir;
```

rentrer_pass

La passerelle:

- Doit rentrer si le tramway a envie de partir.
- Doit attendre que la porte soit fermée avant de rentrer.

Cette spécification donne comme code Lustre:

```
rentrer_pass = pass_sortie and vapartir and not porte_ouverte;
```

porte_pass_ok

La porte et la passerelle sont "OK" si le tramway voulait partir et que la porte a fini de fermer et la passerelle est rentrée.

Cette spécification donne comme code Lustre:

```
porte_pass_ok = vapartir and not( porte_ouverte or pass_sortie );
```

Tests et preuves

Tests

Nous avons effectué plusieurs tests, par exemple un d'entre eux se déroule de la façon suivante:

- Toutes les entrées sont à faux, donc le tramway est en mouvement.
- On demande la porte. Le tramway met **porte_demandee** à vrai.
- On attend un peu, et demande la passerelle. Le tramway met **pass_demandee** à vrai.
- Maintenant, on arrête le tramway en mettant **en_station** à vrai. Le contrôleur agit de façon attendue en essayant de sortir la passerelle. Pendant ce temps, la sortie **porte_demandee** reste toujours à vrai (car la porte attend la passerelle).
- Au bout de quelques cycles, on met **pass_sortie** à vrai et la contrôleur commence à ouvrir la porte.
- On met **porte_ouverte** à vrai. Le contrôleur arrête d'émettre **ouvrir_porte**.
- Au bout de quelques cycles, on signale **attention_depart**. Le contrôleur commence à fermer la porte.
- Une fois que l'on a mis **porte_ouverte** à faux, le contrôleur commence à rentrer la passerelle.
- Finalement, quand on met **pass_sortie** à faux, le contrôleur émet **porte_pass_ok** jusqu'à ce que le tramway se mette en mouvement (**en_station** devient faux).

Le contrôleur semble répondre de façon correcte.

Preuves

Assertions

Nous avons trois assertions essentiels sur le fonctionnement du système:

(1) La porte ne s'ouvre que si le contrôleur l'a demandé:

```
((not pre porte_ouverte and porte_ouverte) => ouvrir_porte)
```

(2) La passerelle ne sort que si le contrôleur l'a demandé

```
((not pre pass_sortie and pass_sortie) => sortir_pass)
```

(3) Le tramway attend au moins un tick avant de signaler **attention_depart**:

```
((not pre en_station and en_station) => not attention_depart)
```

D'autres assertions sont aussi utilisées pour certains preuves. Elles seront introduites quand c'est nécessaire.

Contrôleur stable

Le contrôleur est dit “stable” s'il n'envoie pas des ordres contradictoires: il n'essaye pas d'ouvrir et de fermer la porte en même temps, de même n'essaye pas de sortir et de rentrer la passerelle en même temps.

```
#(ouvrir_porte, fermer_porte)
and
#(sortir_pass, rentrer_pass)
```

Cette preuve se fait sans aucune assertion.

Passagers vivants et tramway non-cassé

Les passagers meurent (cessent d'être vivants) du moment que la porte du tramway s'ouvre quand ce dernier n'est pas en station. Similairement, le tramway casse si la passerelle est sortie alors que le tramway n'est pas en station.

```
not((porte_ouverte or pass_sortie) and not en_station)
```

Cette preuve portant sur l'état de la porte et de la passerelle, les assertions (1) et (2) sont utilisées.

Confort passagers (1)

Les passagers sont confortables si la porte s'ouvre quand ils le demandent et que le tramway est en station. Donc:

```
(en_station and demande_porte and not porte_ouverte) => ouvrir_porte
```

Pour prouver ceci, les assertions suivantes sont nécessaires:

- **attention_depart** n'a pas été signalé

```
not attention_depart
```

- La passerelle n'est pas en train de sortir

```
not sortir_pass
```

Confort passagers (2)

Les passagers sont encore plus confortables si le système se rappelle qu'ils avaient demandés à descendre pendant que le tramway roule et que les portes s'ouvrent une fois l'arrêt atteint.

```
(en_station and pre porte_demandee) => ouvrir_porte
```

Pour prouver ceci, les assertions suivantes sont nécessaires:

- L'assertion (1)
- L'assertion (3)
- **attention_depart** n'a pas été signalé

```
not attention_depart
```

Confort passagers (3)

Ce critère de confort s'adresse aux passagers qui désirent utiliser la passerelle: si un usager avait demandé la passerelle, alors la passerelle sort et la porte s'ouvre une fois que le tramway atteint son arrêt.

```
((en_station and pre pass_demandee) => sortir_pass)
and
((en_station and pass_sortie) => ouvrir_porte)
```

Pour prouver ceci, les assertions (1), (2) et (3) sont nécessaires.

Attention chevilles

La passerelle ne doit pas bouger (ni rentrer, ni sortir) si la porte est ouverte.

```
not(porte_ouverte and (sortir_pass or rentrer_pass))
```

Cette preuve se fait sans aucune assertion.

La porte ferme et la passerelle rentre

Une fois que le signal attention_depart est émis, la porte doit fermer et si nécessaire la passerelle doit rentrer.

```
((porte_ouverte) => fermer_porte)
and
((pass_sortie and not porte_ouverte) => rentrer_pass)
```

Dans ce cas, nous utilisons les assertions comme cas de départ:

```
en_station and attention_depart
```

Le départ est possible

Une fois que le départ devient possible (donc la porte est fermée et la passerelle rentrée), le contrôleur signale bien que la porte est la passerelle sont “OK”:

```
porte_pass_ok
```

Dans ce cas, nous utilisons les assertions comme cas de départ:

```
en_station and attention_depart and not(porte_ouverte or pass_sortie)
```

Conclusions

Un contrôleur de porte de tramway avec passerelle qui fonctionne de façon attendu (suit les spécifications) a donc été implémenté avec succès.