Chapter 1

TraficLights-student

Nous allons utiliser la méthode du formalisme B pour spécifier un problème de gestion des feux d'un carrefour. Cet exemple est largement inspiré d'une petite étude de cas que J.-Y. Chauvet a publiée dans '1st Conference on the B method, Proceedings, ed. Henri Habrias, Nantes 1996'.

1.1 Spécification

Le système se compose de deux feux tricolores dont les couleurs sont vert, orange ou rouge.

Le système a deux états: hors service ou en service.

Le système implémente deux opérations: mise en service et changement de couleur des feux.

Lorsque le système est hors service, les deux feux sont oranges.

Lorsque le système est en service, un et un seul des deux feux est au rouge.

Les feux changent de couleur selon le cycle suivant: orange, rouge, vert, orange.

1.2 Formalisation

La spécification du système et ses obligations de preuve sont rassemblées dans le module *TraficLights*.

`Module` *TraficLights*.

1.2.1 Contexte du système

Le contexte du système définit l'ensemble des couleurs: «les couleurs sont vert, orange et rouge»

```
Module Context.
  Inductive color := red | yellow | green.
End Context.
```

```
Import Context.
```

1.2.2 Etat du système

L'état du système doit permetre de déterminer si celui-ci est «en service ou hors service» et de connaître la couleur des «deux feux tricolores».

```
Record State : Set := mkState {
                        service : bool;
                        light1 : color;
                        light2 : color
                      }.
```

Pour alléger les écritures, on définit trois opérations d'affection de l'état

```
Definition set_service (b:bool) (st:State) :=
  (mkState b st.(light1) st.(light2)).

Definition set_light1 (c:color) (st:State) :=
  (mkState st.(service) c st.(light2)).

Definition set_light2 (c:color) (st:State) :=
  (mkState st.(service) st.(light1) c).
```

1.2.3 Invariants du système

Système hors service: «lorsque le système est hors service, les deux feux sont oranges»

```
Definition Inv_1 (st:State) : Prop :=
  (*  A DEFINIR *) .
```

Système en service: «Lorsque le système est en service, un et un seul des deux feux est au rouge.

```
Definition Inv_2 (st:State) : Prop :=
  (*  A DEFINIR *) .
```

1.2.4 Initialisation du service

La «mise en service» est réalisée dans le module *Init*.

```
Module Init.
```

NB: l'initialisation du système est inconditionnelle

```
Definition Guard (st:State) := True.
```

Transition de mise en service

```
Definition action (st:State) :=
   A DEFINIR  .
```

Obligations de preuve

```
Theorem PO_Safety_1 : (*  A DEFINIR *) .
Proof.
  (*  A FAIRE *)
Qed.
Theorem PO_Safety_2 : (*  A DEFINIR *) .
Proof.
  (*  A FAIRE *)
Qed.
End Init.
```

1.2.5 Changement de couleur

```
Module ChangeColor.
```

Guarde: le système est en service

```
Definition Guard (st:State) := (*  A DEFINIR *) .
```

Transition de changement de couleur: «Les feux changent de couleur selon le cycle suivant: orange, rouge, vert, orange»

```
Definition action (st:State) := (*  A DEFINIR *) .
```

Propriété utile du changement de couleur: le champ *service* n'est pas affecté par la tarnsition.

```
Lemma change_service :
  ∀ (st:State),
    (action st).(service) = st.(service).
Proof.
  intro st.
  unfold action.
  destruct (light1 st).
  - (*  light1 green *)
    destruct (light2 st) ; auto.
  - (*  light1 red *)
    simpl. reflexivity.
  - (*  light1 yellow *)
    simpl. reflexivity.
Qed.
```

Obligations de preuve

```
Theorem PO_Safety_1 : (*  A DEFINIR *) .
```

```
    Proof.
      (*  A FAIRE *)
    Qed.

    Theorem PO_Safety_2 : (*  A DEFINIR *) .
    Proof.
      (*  A FAIRE *)
    Qed.

  End ChangeColor.

End TraficLights.
```