Programmation synchrone – Master 2 LC
TP 2 : Machines à état en SCADE

Mehdi Dogguy
http://www.pps.jussieu.fr/~dogguy/?page=teaching/progsyn/

Sur chaque machine, il existe une documentation (*User Manual*) de l'outil SCADE (Démarrer → Tous les Programmes → SCADE → Documentation).

Exercice 1 : *Échauffement*

Il s'agit de *(re)programmer et simuler* les exemples vus en cours et quelques exemples supplémentaires.

1. *Détection de front montants :* (edge)
   - Entrée : x (booléen).
   - Sortie : y (booléen).
   - Fonction : y_t vrai ssi x_{t-1} existe et est faux et x_t est vrai.

   Remarques :
   - Les fronts descendants de x sont simplement les fronts montants de la négation de x.
   - Il s'agit d'un détecteur de fronts strict, c'est-à-dire que si x est vrai à l'instant initial, on ne considère pas cela comme un front. Comment peut-on réutiliser edge pour avoir un front montant avec l'instant initial ?
2. *Compteur :*
   - Entrées : inc (entier), init (entier), reset (booléen)
   - Sortie : N (entier)
   - Fonction : N est init quand reset est vrai, sinon il est incrémenté de inc à chaque tic de l'horloge.

   Réutilisez ce compteur pour :
   - énumérer les nombres paires
   - énumerer en cycle les entiers modulo 5
3. *Mémoire bien initialisée :* (jafter)
   - Entrée : x, booléen
   - Sortie : y, booléen
   - Fonction : y_t vrai ssi x_{t-1} existe et est vrai.
4. *Automate à deux états :* (switch)
   - Entrées : orig, on, off (booléens).

- Sortie : state (booléen).
- Fonction : state passe de faux à vrai sur la commande on, de vrai à faux sur la commande off. Tout doit se passer comme si state était égal à orig avant le premier instant.

Exercice 2 : *Montre digitale simple*

Une montre digitale simple a que deux boutons `start_stop` et `reset` (modélisés par des variables booléennes). Elle reçoit également l'évènement d'écoulement de 1/100 secondes par une entrée (booléenne) `hs`. Elle calcule :

- sur `running` (booléen) son état
- sur `time` (entier) le nombre de `hs` reçus tant qu'il est `running` depuis le dernier `reset`

Vous pouvez écrire une solution directement ou en utilisant le noeud compteur.

Exercice 3 : *Montre digitale élaborée*

Une montre digitale élaborée est une montre digitale (comme ci-dessus) qui permet en plus d'enregistrer un temps intermédiaire. Elle gère deux temps :

- un temps interne `internal_time` calculé comme précédemment,
- un temps intermédiaire `displayed_time` qui reste constant quand la montre est "gelée" et qui est égal à `internal_time` sinon.

Le bouton `reset` est utilisé pour changer l'état "gelé/non gelé" de l'affichage :

- initialement, l'état est non gelé,
- si `reset` arrive et la montre est `running` et pas gelée, elle devient gelée,
- si `reset` est appuyé quand la montre est gelé, elle devient non gelée.

Le bouton `reset` est interprété comme une remise à zéro uniquement quand la montre est arrêtée et pas gelée.

Exercice 4 : *Déjà vu — Carrefour*

Un carrefour à deux voies orthogonales (EstWest et NordSud) est contrôlé par deux paires de feux tricolores (rouge, jaune, vert).

1. Modéliser en flots de données un seul feux tricolore (français) :
   - Entrées : reset (booléen)
   - Sorties : red, yellow, green (booléens)
   - Fonction : reset remet le feu à rouge, sinon les sorties deviennent vraies (exclusivement) dans l'ordre red, green, yellow, red, ...

   On considère que la durée de chaque feu est la durée de l'horloge.
2. Composer deux feux ainsi écrits afin d'obtenir un carrefour.

Correction 1 : *Echauffement*

1. *Mémoire bien initialisée :* (jafter)

```
node jafter (x: bool) returns (y: bool);
let
  y = false -> pre(x);
tel
```

2. *Détection de front montants :* (edge)

```
node edge (x: bool) returns (y: bool);
let
  y = 0 -> x and not(pre(x))
tel

node nedge (x: bool) returns (y: bool);
let
  y = edge(not(x));
tel

node iedge (x: bool) returns (y: bool);
let
  y = x -> edge(x);
tel
```

3. *Automate à deux états :* (switch)

```
node switch (orig, on, off: bool) returns (state: bool);
let
  state0 = orig -> pre(state);
  state = if on and not state0 then false
          else if off and state0 then true
          else state0;
tel
```

4. *Compteur :*

```
node Counter(init,incr: int; reset: bool) returns (N: int);
let
  N = init -> if reset the init
```

```
                    else pre(N) + incr;
    tel
    even = COUNTER(0,2,false);
    modulo5 = COUNTER(0,1,(pre(modulo5)=4));
```

Correction 2 : *Montre digitale simple*

```
node Simple_Stopwatch(start_stop, reset, hs: bool)
    returns (time: int; running: bool);
let
  time = 0 -> if hs and running the pre(time) + 1
              else if reset then 0 else pre(time)
(*time = condact(Counter((0,1,reset) when clock));
  clock = (hs and running) or (true -> reset);
 *)
  running = false -> if start_stop then not pre(running)
                     else pre(running);
tel
```

Correction 3 : *Montre digitale élaborée*

```
node Stopwatch(start_stop, reset, hs: bool)
    returns (displayed_time: int; running, frozen: bool);
var internal_time: int; actual_reset: bool;
let
  frozen = false -> if reset and pre(running) then true
                    else if reset and pre(frozen) then false
                    else pre(frozen);
  displayed_time = condact(internal_time,frozen);
  (internal_time, running) = Simple_Stopwatch(start_stop, actual_reset, hs);
  actual_reset = reset and pre(not running and not frozen);
tel
```

Correction 4 : *Déjà vu — Carrefour*

```
node TrafficLight(rinit, yinit, ginit: bool)
    returns (red, yellow, green: bool);
let
  assert (# (rinit, yinit, ginit));
  red = rinit -> pre(yellow);
  yellow = yinit -> pre(green);
  green = ginit -> pre(red);
tel
node Intersection (rg: bool) returns
    returns (redEW, yellowEW, greenEW, redNS, yellowNS, greenNS: bool);
let
  (redEW, yellowEW, greenEW) = TrafficLight(rg,false,not(rg));
  (redNS, yellowNS, greenNS) = TrafficLight(not(rg),false,rg);
tel
```