Prism… rappels

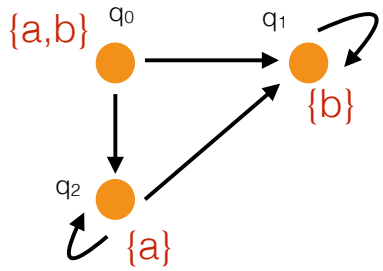
Problèmes de vérification pour LTL

Prism

Model-checking

- $\mathbf{S} \models \phi$?

$\mathbf{S}$



$\phi = \mathbf{G}\,(a \Rightarrow \mathbf{F}\,b)$

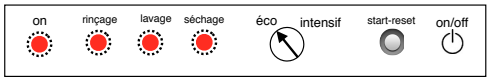
```
mdp

label "a" = (q=0 | q=2);
label "b" = (q=0) | (q=1);

module K
        q : [0..2] init 0;
    [] q=0 -> (q'=1);
    [] q=0 -> (q'=2);
    [] q=1 -> true;
    [] q=2 -> true;
    [] q=2 -> (q'=1);
endmodule
```

```
A [ G (« a » => (F « b »))]
```

On s'intéresse ici à la spécification en LTL d'un lave-vaisselle. Le tableau de bord de la machine comporte quatre voyants lumineux (« on », « rinçage », « lavage », « séchage ») un sélecteur de programme avec deux positions (« éco » et « intensif »), et deux boutons (« start-reset » et « on/off »). La figure 1 représente ce tableau de bord.

On considère les propositions atomiques suivantes : chaque voyant a une proposition atomique associée qui est vraie lorsque le voyant est allumé (Von, Vrinçage, Vlavage, Vséchage), Péco est vrai lorsque le sélecteur de programme est sur « éco » (et Pintensif est vrai lorsqu'il est sur « intensif »). Bstart-reset est vrai au moment où le bouton « start-reset » est enfoncé, et Bon/off est vrai lorsque c'est le bouton « on/off » qui est enfoncé.

La machine peut se trouver dans deux **états** différents : allumé (proposition Pon) ou éteinte (proposition Poff). Lorsqu'elle est allumée, elle peut être dans trois **modes** différents : attente, marche, ou fin. A chacun de ces modes on associe une proposition atomique : attente, marche, fin.



Écrire les formules suivantes :

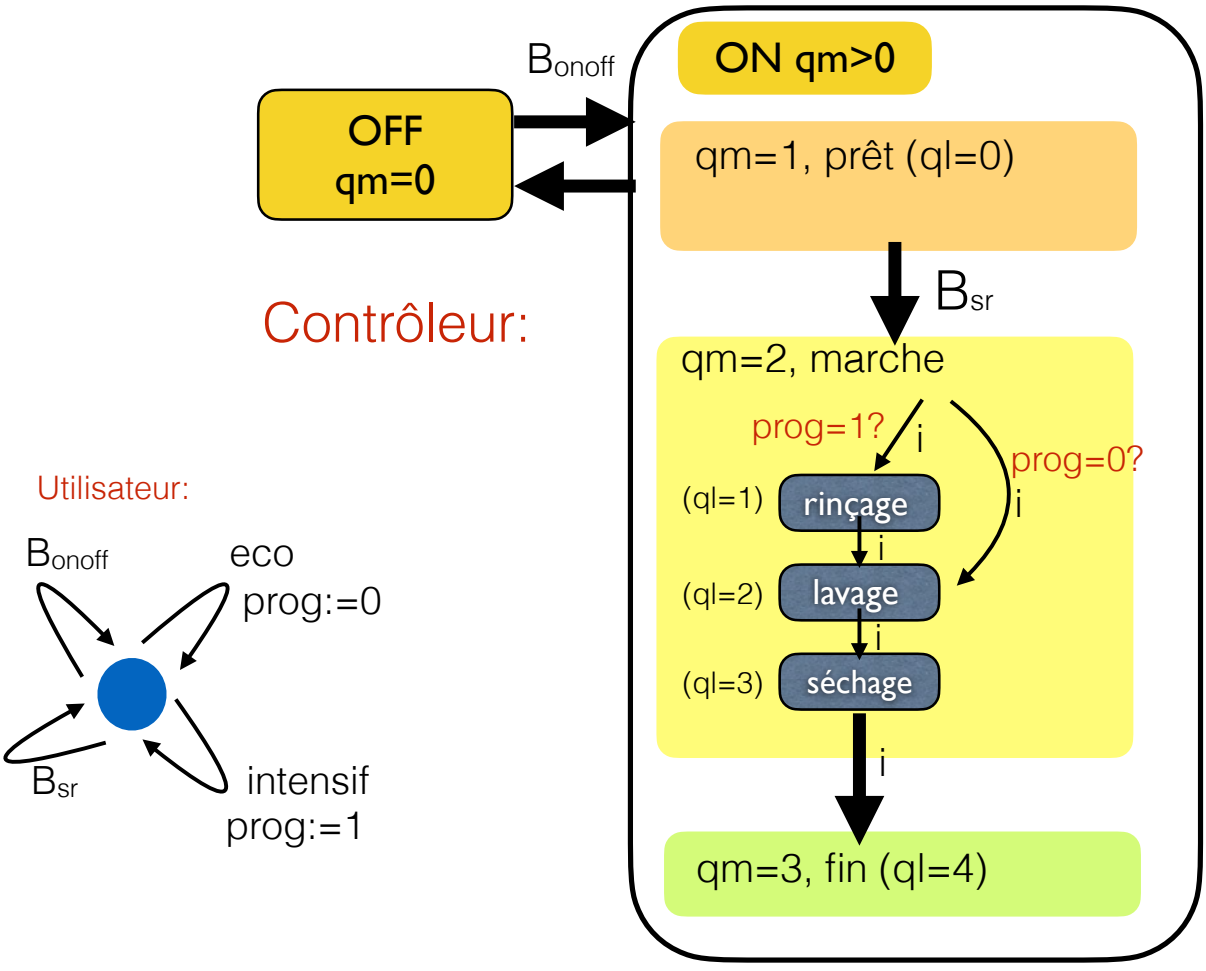
1. Une formule exprimant que la machine est toujours dans un seul état (allumée ou éteinte). Ecrire une formule qui spécifie que le voyant Von est allumé si et seulement si la machine est allumée.
2. Une formule pour indiquer que l'état de la machine change avec le bouton Bon/off (appuyer sur le bouton la fait changer d'état à l'instant suivant).
3. Une formule qui spécifie qu'à chaque instant, si la machine est allumée, alors elle est dans un et un seul des trois modes (attente, marche, fin). Et une formule pour spécifier qu'à l'allumage, elle est en attente.
4. Une formule qui énonce qu'appuyer sur le bouton Bstart-reset lorsque la machine est en attente, la fait passer dans le mode marche.

5. Et une formule pour dire que si le bouton Bstart-reset est appuyé longtemps (= pendant 3 instants successifs), alors la machine passe en mode attente (quel que soit le mode dans lequel elle était auparavant).
6. Une formule $\phi(V)$ qui indique qu'un voyant V clignote (ici la proposition V est vraie ssi le voyant est allumé) : c'est-à-dire qu'il est allumé, puis éteint, puis allumé, *etc.* changeant à chaque instant (ou éteint puis allumé, puis éteint...). On supposera que ce clignotement ne s'arrête qu'avec l'arrêt de la machine (état Poff).
7. Une formule qui énonce que les voyants « rinçage », « lavage » et « séchage » clignotent lorsque la machine arrive dans le mode « fin ».
8. Une formule spécifiant qu'à tout moment un programme (et un seul) est sélectionné.

9. Une formule qui spécifie que lorsque la machine passe en mode « marche » et que le programme sélectionné est « intensif », alors elle va exécuter un cycle de lavage complet (c'est-à-dire que le voyant « rinçage » s'allumera pendant une certaine période, puis ce sera au tour du voyant « lavage » puis au tour du voyant « séchage ») à moins de s'interrompre en cas de retour au mode « attente » ou à l'état « éteint ».
10. Une formule qui spécifie que lorsque la machine passe en mode « marche » et que le programme sélectionné est « éco », alors elle va exécuter un cycle de lavage sans rinçage (c'est-à-dire que le voyant « lavage » s'allumera pendant une certaine période, puis ce sera au tour du voyant « séchage ») à moins de s'interrompre en cas de retour au mode « attente » ou à l'état « éteint ».

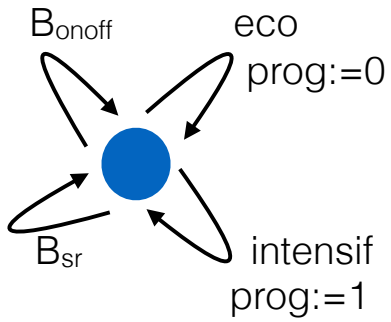
Un modèle pour le LV

fichiers:
lv.prism
lvprop



```
module Utilisateur
    prog : [0..1] init 0; // 0 : eco 1: intensif
    bouton : bool init false; // passe de F a V ou de V a F quand
                              une action externe a lieu
 [BOnOff] true -> (bouton'=!bouton);
 [Bsr] true -> (bouton'=!bouton);
 [eco] true -> (prog'=0) & (bouton'=!bouton);
 [intensif] true -> (prog'=1) & (bouton'=!bouton);
endmodule
```

Utilisateur:




```
module Controleur                                   var d'état
    qm : [0..3] init 0;  // 0:off 1:attente 2:marche 3: fin
    ql : [0..4] init 0;  // 0:pret, 1: rincage, 2:lavage, 3:séchage, 4:fin
    cpteurbsr : [0..2] init 0; // compte le nb de pressions successives sur Bsr
    bonoff : bool init false;
    bsr : bool init false;                          <- var aux.
    von : bool init false;
    vrincage : bool init false;
    vlavage : bool init false;                      Voyants
    vsechage : bool init false;

    [BOnOff] (qm=0) -> (qm'=1) & (von'=true) & (vrincage'=false) &
                       (vlavage'=false) & (vsechage'=false) &
                       (cpteurbsr'=0) & (bonoff'=true) & (bsr'=false);

    [BOnOff] (qm>0) -> (qm'=0) & (ql'=0) & (von'=false) &
                (vrincage'=false) & (vlavage'=false) &
                (vsechage'=false) & (cpteurbsr'=0) & (bonoff'=true) &
                (bsr'=false);
...
```

module Controleur (suite)

```
[Bsr] (qm=1) & (cpteurbsr=0) -> (qm'=2) & (cpteurbsr'=1) &
                                (bsr'=true)& (bonoff'=false);

[Bsr] (qm>0) & (cpteurbsr=1) -> (cpteurbsr'=2) & (bsr'=true) &
                                (bonoff'=false);

[Bsr] (qm=2) & (cpteurbsr=0) -> (cpteurbsr'=1) & (bonoff'=false) &
                                (bsr'=true);

[Bsr] (qm=3) & (cpteurbsr=0) -> (cpteurbsr'=1) & (bonoff'=false) &
                                (bsr’=true);

[eco] (true) -> (cpteurbsr'=0) & (bsr'=false) & (bonoff’=false);

[intensif] (true) -> (cpteurbsr'=0) & (bsr'=false) & (bonoff’=false);
```

…

module Controleur (suite)

```
[Bsr] (qm=0) -> (qm'=0) & (bsr'=true) & (bonoff'=false) ;
      [Bsr] (qm>0) & (cpteurbsr=2) -> (qm'=1) & (ql'=0) &
(cpteurbsr'=0) & (bsr'=true)& (bonoff'=false);
      [Bsr] (qm=1) & (cpteurbsr=0) -> (qm'=2) & (cpteurbsr'=1) &
(bsr'=true)& (bonoff'=false);
      [Bsr] (qm>0) & (cpteurbsr=1) -> (cpteurbsr'=2) & (bsr'=true) &
(bonoff'=false);
      [Bsr] (qm=2) & (cpteurbsr=0) -> (cpteurbsr'=1) & (bonoff'=false)
& (bsr'=true);
      [Bsr] (qm=3) & (cpteurbsr=0) -> (cpteurbsr'=1) & (bonoff'=false)
& (bsr’=true);
```

qm : [0..3] init 0; // 0:off 1:attente 2:marche 3: fin

```
module Controleur (suite)

    [i] (ql=0) & (qm=2) & (prog=0) -> (ql'=2) & (vlavage'=true) &
                                      (bonoff'=false) & (bsr'=false) &
                                      (cpteurbsr'=0); // debut eco
    [i] (ql=0) & (qm=2) & (prog=1) -> (ql'=1) & (vrincage'=true) &
                                      (bonoff'=false) & (bsr'=false) &
                                      (cpteurbsr'=0); // debut intensif
    [i] (ql=1) & (qm=2) -> (ql'=2) & (vrincage'=false) & (vlavage'=true)
                           & (bonoff'=false) & (bsr'=false) & (cpteurbsr'=0);
    [i] (ql=2) & (qm=2) -> (ql'=3) & (vsechage'=true) & (vlavage'=false)
                           & (bonoff'=false) & (bsr'=false) & (cpteurbsr'=0);
    [i] (ql=3) & (qm=2) -> (ql'=4) & (qm'=3)  & (vrincage'=true) &
                (vlavage'=true) & (vsechage'=true) & (bonoff'=false) &
                (bsr'=false) & (cpteurbsr'=0);
    [i] (ql=4)   -> (vrincage'=!vrincage) & (vlavage'=!vlavage) &
                 (vsechage'=!vsechage) & (bonoff'=false) &
                 (bsr'=false) & (cpteurbsr'=0);
endmodule
```