Cours MOdélisation, Vérification et EXpérimentations
Exercices
Modélisation et vérification avec LUSTRE et KIND2 (I)
par Dominique Méry
14 janvier 2026

Outils à récupérer

- `https://www-verimag.imag.fr/DIST-TOOLS/SYNCHRONE/reactive-toolbox/` est la page `Development Tools for Critical Reactive Systems using the Synchronous Approach - The Verimag Reactive Tool Box` qui donne accès à des outils, pour traiter les programmes réactifs en LUSTRE.
  Cette page propose un ensemble très pertinent d'outils développés à Grenoble et en particulier dans le laboratoire Verimag. Vous pouvez récupérer `Lustre V6` pour compiler les prorammes LUSTRE.
- `https://kind2-mc.github.io/kind2/` est la page de l'outil Kind2 permettant de vérifier des propriétés des programmes LUSTRE. Vous pouvez installer cet outil sur votre PC mais vous pouvez aussi utiliser l'interface Web qui permet de réaliser certaines opérations de vérification, de simulation ou d'analyse.mini

Le langage utilisé par Kind2 est un sous-langage de LUSTRE et il n'y a pas les éléments de manipulations des horloges.

Exercice 1 *(Utilisation d'exemples simples)*
Pour chaque exemple, il faut utiliser l'interface web et comprendre la syntaxe et la sémantique des programmes LUSTRE.

Question 1.1 *Expliquer simplement ce que calcule les fonctions ou nœuds.*

Calculs

Listing 1 – operations.lus

```
--  operations over nodes
node op (X: int;Y:int) returns (S,P,POLY,POLY2:int;TEST:bool);
let

   S =  X + Y;
   P = X*Y;
   POLY = X*X + 2*X*Y + Y*Y;
   POLY2 = S*S;
   TEST = (POLY = POLY2);
   check TEST;

tel
```

Question 1.2 *Expliquer le calcul de la fonction de fibonacci.*

Calcul Fibonacci

Listing 2 – fibo.lus

```
--------------------------------------------------------------------
-- This node produces the Fibonacci series: 1,1,2,3,5,8,13,21,...
--------------------------------------------------------------------
```

```
node fibo(_:bool) returns(Fib: int);
let
   Fib = 1 -> pre (1 -> Fib + pre Fib);
tel
```

Question 1.3 *La fonction power2 est définie avec des suites particuli'eres. Définissez les suites et exp)liquez pourquoi elles calculent la fonctioon power2.*

Power 2

Listing 3 – power2.lus

```
-- power2(_) contains all the square  numbers in increasing order
-- where 0^2 = 0
--       (n+1)^2 = n^2 +2*n + 1
node power2(_ : bool) returns (P: int);
var W: int;
let
  -- all the natural even numbers
  W = 1 -> (pre W) + 2;
  P = 0 -> (pre P) + (pre W);

tel
```

Exercice 2 *On s'inspire du calcul de la puissance de 2 pour calculer la puissance de 3 :*

```
int power3(int x)
{int cz,cv,cw,ct,k,r;
  cz=0;cv=0;cw=1;ct=3;k=0;
while (k<x)
        {
          cz=cz+cv+cw;
          cv=cv+ct;
          ct=ct+6;
          cw=cw+3;
          k=k+1;
          };
  r=cz;
  return (r);
}
```

Ecrire un programme LUSTRE qui réalise le calcul du flôt des valeurs de la puissance de 3 et qui vérifie avec la directive check que la fonction est correcte.

Exercice 3 *Soit la fonction puissance 4.*

Question 3.1 *Utilisez le nœud* `power2` *pour construire un nœud calculant la suite des valeurs des puissances de 4.*

Question 3.2 *En utilisant un observateur, montrer que la suite est correctement formée des puissances de 4.*

Exercice 4
- *Ecrire un programme LUSTRE calculant la somme des n premiers entiers positifs.*
- *Montrer que votre solution est correcte.*

Cours MOdélisation, Vérification et EXpérimentations
Exercices
Modélisation et vérification avec LUSTRE et KIND2 (II)
par Dominique Méry
14 janvier 2026

☛ *La méthodologie de conception est fondée sur la description des flux d'(entrées et de sorties et sur la modélisation de lénvironnement. Les système est vu comme un tout. Puis on reprend la liste des exigences ou requirements qui sont soumises au système.*

Exercice 5 *Les feux tricolores doivent vérifier des propriétés de sûreté et nous proposons ce système de feux tricolores.*

trafficlight

Listing 4 – trafficlight.lus

```
node TrafficLight( Button: bool )
returns ( Red, Yellow, Green, Walk, DontWalk: bool );

var Phase, prePhase: int;
let
  prePhase = 0 -> pre Phase;
  Phase    = if Button then
               1
             else if 0 < prePhase and prePhase < 10 then
               prePhase + 1
             else
               0;

  Green    = Phase = 0;
  Yellow   = Phase = 1;
  Red      = Phase > 1;

  Walk     = Phase > 2 and Phase < 10;
  DontWalk = not Walk;
tel
```

Question 5.1 *Exprimer léxigence suivante : Walk n'est jamais vrai quand les voitures sont autorisées*

Question 5.2 *Red et Green ne sont jamais vraie en même temps.*

Question 5.3 *Red ne peut pas suivre immédiatement de Green*

☛ *Il peut être nécessaire de définir des définitions auxiliaires.*

Exercice 6 *On considère un interrupteur* `Switch` *simplifié et on propose d'analyser le programme. Des définitions auxiliaires sont indiquées.*

Switch simplifié

Listing 5 – simpleswitch.lus

```
------------------------------------------------------------
-- Boolean Switch
--
```

```
-- Model a switch with two buttons, Set and Reset.
-- Pressing Set turns the switch on.
-- Pressing Reset turns the switch off
-- If Set and Reset are initially both unpressed,
-- the initial position of the switch is determined by
-- a third signal, Init

node Switch( Set, Reset, Init : bool ) returns ( X : bool );
let
    X =      if Set then true
        else if Reset then false
        else (Init -> pre X);
tel

--------------------------------------------------------------------
-- The following observer expresses 3 safety requirements
-- for the switch.

--  R1: Setting turns the switch on
--  R1: Resetting turns the switch off
--  R3: Doing nothing keeps the switch as it was
--  R4: The reset signal is ignore when Set is true
--  R5: If the switch is on, it stays so until the next reset
--  R6: If the switch is off, it stays so until the next set
--  R7: Never setting or resetting maintains the initial position


-- Notes
--   - In R4 "ignoring" means that the behavior of Switch is exactly the same,
--     independently of the value of Reset when Set is true.
--   - R5 is equivalent to: if we have not reset, since the last time we set
--     then the switch is on
--   - R6 is equivalent to: if we have not set, since the last time we reset
--     then the switch is off


-------------------------------
-- Auxiliary temporal operators
-------------------------------

-- First(X) is the constant stream consisting of the first value of X
node First( X : bool ) returns ( Y : bool );
let
  Y = X -> pre Y;
tel

-- Sofar(X) is true at any point iff X has been true from the beginning
-- until that point
node Sofar( X : bool ) returns ( Y : bool );
let
  Y = X -> (X and (pre Y));
tel

-- Since(X,Y) is true precisely when X has been true at some point and
-- Y has been continuously true afterwards
node Since( X, Y : bool ) returns ( Z : bool );
```

```
let
  Z = X or (Y and (false -> pre Z));
tel
```

Les exigences suivantes sont à traduire et à ajouter dans le module `reqsimpleswitch.lus`.

```
    -- R1: Setting turns the switch on
-- R1: Resetting turns the switch off
-- R3: Doing nothing keeps the switch as it was
-- R4: The reset signal is ignore when Set is true
-- R5: If the switch is on, it stays so until the next reset
-- R6: If the switch is off, it stays so until the next set
-- R7: Never setting or resetting maintains the initial position
```

Cours MOdélisation, Vérification et EXpérimentations
Exercices
Modélisation et vérification avec LUSTRE et KIND2 (III)
par Dominique Méry
14 janvier 2026

Exercice 7 *Soit la suite N2 des entiers naturels commençant par 2 et la suite des valeurs des nombres premiers PP*
$N2 = (2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ \ldots)$
$PP = (2\ 3\ 3\ 5\ 5\ 7\ 7\ 7\ 7\ 11\ 11\ 13\ 13\ 13\ \ldots)$