Le langage LUSTRE : Programmation data-flow synchrone pour les systèmes embarqués

N7 – 3SN

Cours 2: Lustre complet (basic + horloges)

Frédéric Boniol

ONERA - 2, av. E. Belin - 31055 Toulouse

frederic.boniol@onera.fr

Introduction

Première conclusion sur Lustre basic (sans horloge) :

Lustre basic est un langage permettant de décrire (naturellement) des systèmes cycliques :

- déterministes (les équations sont ordonnées de façon unique par l'ordre des flots)
- temps d'exécution borné (pas de processus dynamique, de boucle à longueur variables…)
- mémoire bornée (la profondeur des « pre » est bornée)
- modulaire (un nœud est un opérateur réutilisable sans effet de bord…)

Mais

En l'état, Lustre basic ne permet de concevoir que des systèmes "mono-cycles » (même « sampler » pour tout le monde)

=> extension « multi-cycle »

=> notion d'horloge

1. Lustre avec horloges

Notion d'horloge

Notion d'horloge

Une horloge est un flot booléen.

Horloge de base

On suppose qu'il existe une horloge de base, dénotée par le flot booléen toujours vrai (`true`).

=> L'horloge de base est le signal présent et vrai à chaque réaction du programme. C'est le signal qui caractérise les instants d'activation du programme.

=> Chaque nœud a une horloge de base locale, qui peut être l'horloge de base globale (l'horloge de base du nœud principal) ou une sous-horloge de celle-ci.

Horloge d'un flot

Chaque flot X est typé par une horloge (i.e., un flot booléen).

Un flot X est caractérisé par un couple (V,B) où :

V est la suite infinie ou finie de valeurs v_0, v_1, … v_n, …

B est l'horloge de X, i.e., une suite finie ou infinie de true et de false

=> X est présent avec la valeur v_n au nième instant ou B est vrai, et absent lorsque B est faux ou absent.

Notion d'horloge

Horloge et équation

Les équations doivent être homogènes du point de vue des horloges.

Exemple :

l'équation

```
Z = X + Y
```

n'a de sens que si X et Y ont la même horloge (et le même type), et définit un flot Z de même horloge (et de même type).

=> Toute équation `O=F(I,…)` définit un flot typé, c'est-à-dire :

- une suite de valeurs,
- une horloge, qui doit être construite de façon unique et non ambiguë.

=> Calcul des horloges.

Horloge d'un nœud

=> Un nœud peut recevoir des flots d'horloges différentes.

=> L'horloge d'un nœud (son horloge de base) est l'horloge de son entrée la plus rapide.

Lustre avec horloges

Opérateurs temporels

- opérateur de sous-échantillonnage sur une horloge moins rapide :

  `when`

- opérateur de sur-échantillonnage sur une horloge plus rapide :

  `current`

⇒ when et current sont les deux seuls opérateurs permettant de modifier l'horloge d'un flot.

- opérateur de sous + sur-échantillonnage

  `condact`

  ⇒Construite à partir de **`when`** et de **`current`**

Lustre avec horloges : when

Opérateurs de sous-échantillonnage : `when`

Projette un flot sur une horloge plus lente, permettant ainsi le dialogue d'un processus plus fréquent vers un processus moins fréquent.

Equivalent à un opération de « cast » (changement de type).

Soit le flot X et un flot booléen B (une horloge) de même horloge. L'équation

```
Y = X when B
```

définit un flot Y, de même type que X, et d'horloge B

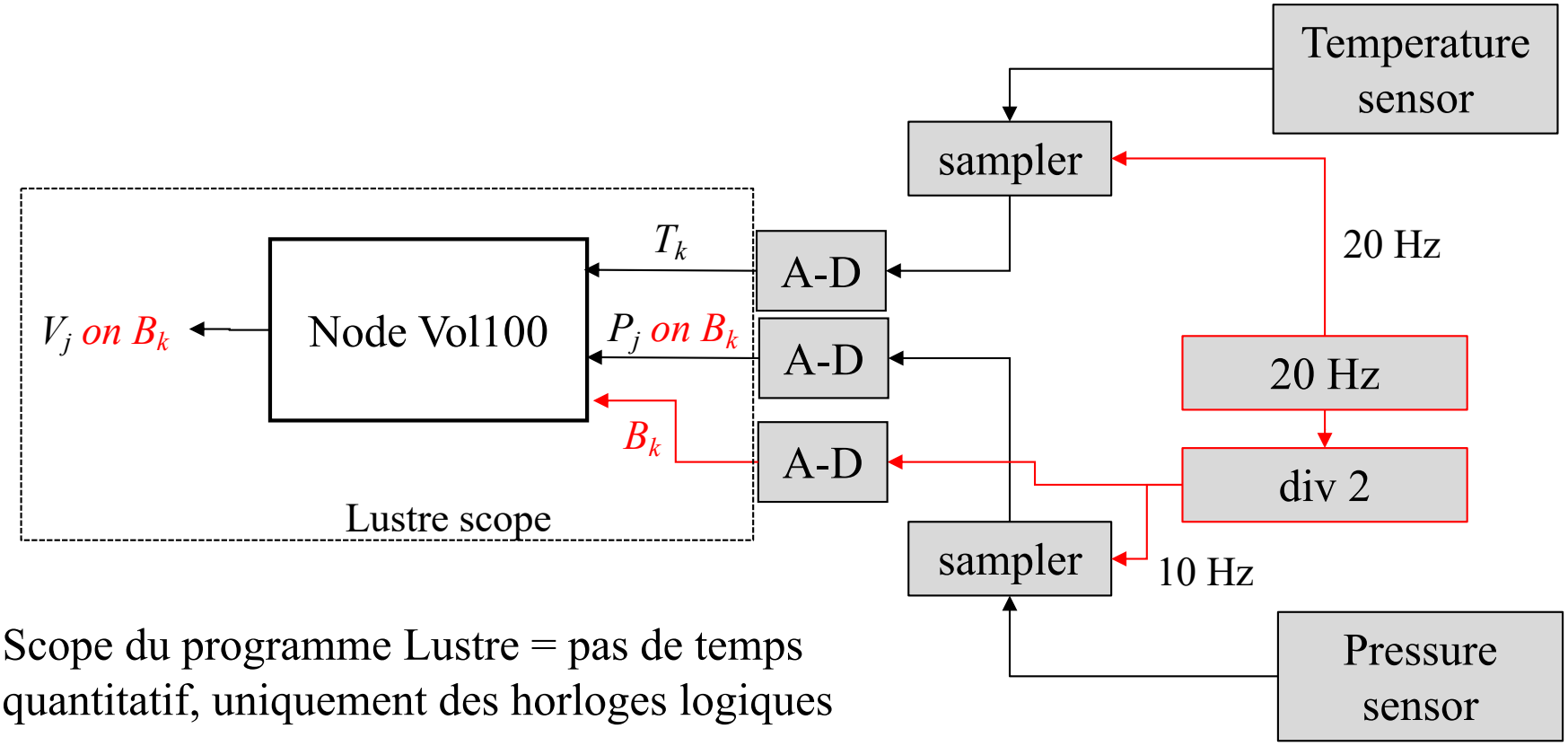
- Y est présent lorsque B est vrai
- Y est absent lorsque B est faux ou lorsque B et X sont absents

X	X0	X1	X2	X3	X4	X5	X6
B	**true**	false	false	**true**	**true**	false	**true**
Y = X when B	X0			X3	X4		X6

Lustre avec horloges : when

Opérateurs de sous-échantillonnage : **when**

Exemple : calcul d'un volume d'une mole de gaz tous les 100ms à partir d'une température et d'une pression arrivant respectivement tous les 50ms (horloge de base) et tous les 100ms :



⇒ Scope du programme Lustre = pas de temps quantitatif, uniquement des horloges logiques

Lustre avec horloges : when

Opérateurs de sous-échantillonnage : **when**

Exemple : calcul d'un volume d'une mole de gaz tous les 100ms à partir d'une température et d'une pression arrivant respectivement tous les 50ms (horloge de base) et tous les 100ms :

```
const R = 8.314;
node VOL100 (T:real; B:bool; P:real when B)
returns (V:real when B)
var T100:real when B;
assert (true -> (B or pre(B));
assert (true -> not (B and pre(B));
let
  V= (T100 / P) * (R when B);
  T100 = T when B;
tel.
```

Lustre avec horloges : current

Opérateurs de sur-échantillonnage : `current`

Projette un flot sur une horloge plus rapide, permettant ainsi le dialogue d'un processus moins fréquent vers un processus plus fréquent.

Soit le flot X et un flot booléen B (une horloge) de même horloge. L'équation

`Y = current X`

définit un flot Y, de même type que X, et d'horloge l'horloge de B

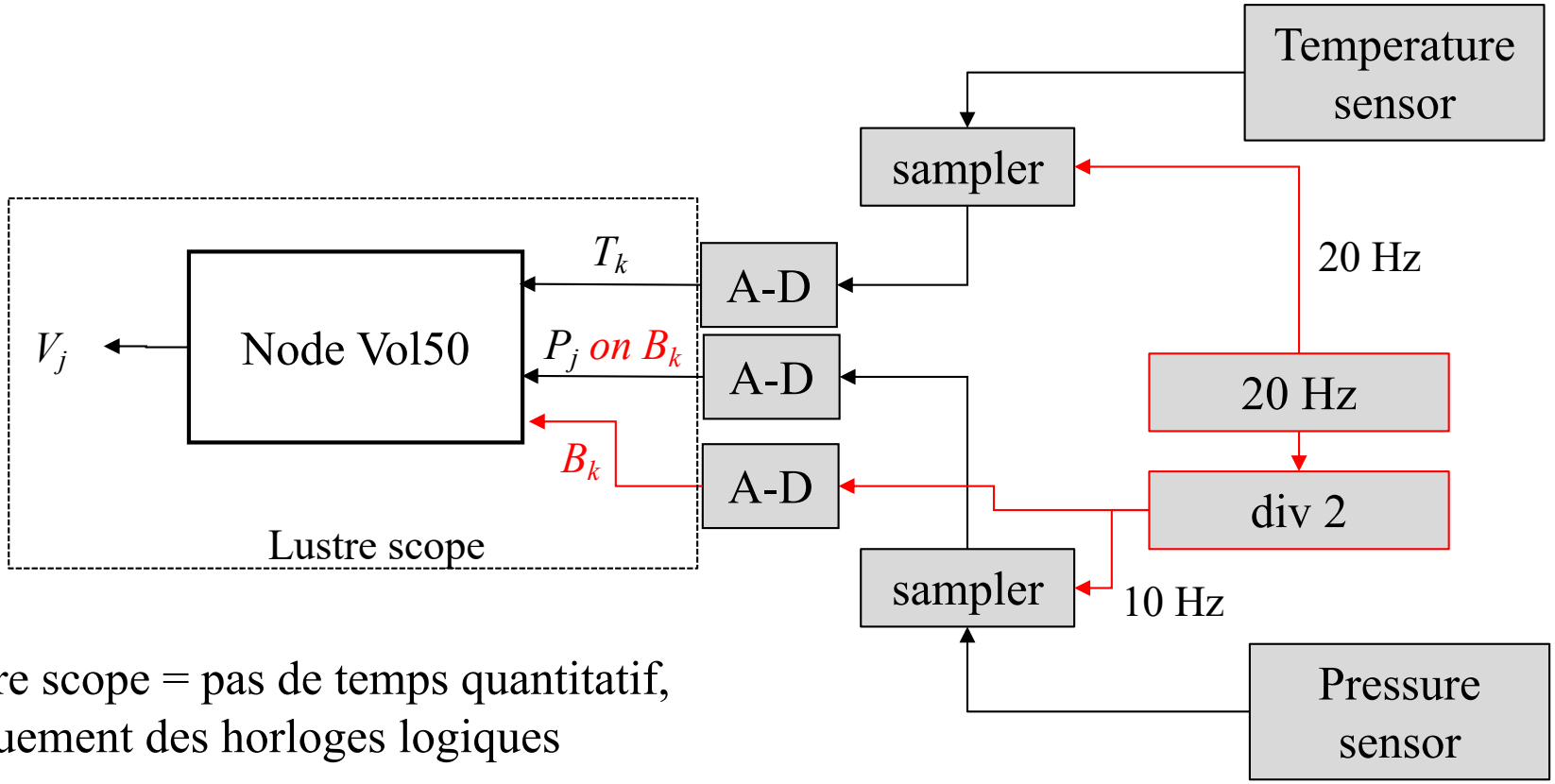
- Y est présent si et seulement si B est présent
- lorsque Y est présent, Y est égal à X si X est présent, sinon à la dernière valeur de X

X	X0	X1	X2	X3	X4	X5	X6
B1	**true**	false	false	**true**	**true**	false	**true**
B2	**true**	**true**	false	false	**true**	**true**	false
X1 = X when B1	X0			X3	X4		X6
B21 = B2 when B1	**true**			false	**true**		false
X21 = X1 when B21	X0				X4		
current (X1)	X0	X0	X0	X3	X4	X4	X6
current (X21)	X0			X0	X4		X4

Lustre avec horloges : current

Opérateurs de sur-échantillonnage : `current`

Exemple : calcul d'un volume d'une mole de gaz tous les 50ms à partir d'une température et d'une pression arrivant respectivement tous les 50ms (horloge de bas) et tous les 100ms :



⇒ Lustre scope = pas de temps quantitatif, uniquement des horloges logiques

Lustre avec horloges : current

Opérateurs de sur-échantillonnage : `current`

Exemple : calcul d'un volume d'une mole de gaz tous les 50ms à partir d'une température et d'une pression arrivant respectivement tous les 50ms (horloge de bas) et tous les 100ms :

```
const R = 8.314;
node VOL50 (T:real; B:bool; P:real when B)
returns (V:real)
var P50:real;
assert (true -> (B or pre(B));
assert (true -> not (B and pre(B));
let
  V= (T / P50) * R;
  P50 = current P;
tel.
```

Lustre avec horloges : exemple

Exemple : une minuterie simple (sur les ticks de base)

Entrée : set	activation de la minuterie (flot booléen)
sortie : level	état de la minuterie (flot booléen)
constante : delay	durée de la minuterie en top de l'horloge de base

```
node stable (set : bool; delay : int)
returns (level : bool)
var count : int;
let
   level = (count > 0);
   count = if set then delay
           else  if (false -> pre(level))
                 then pre(count) - 1
                 else (0 -> pre(count));
tel.
```

(=> voir BE1)

Lustre avec horloges : exemple

Exemple : une minuterie paramétrée par le temps

Entrée :	set	activation de la minuterie (flot booléen)
	second	unité de temps de la minuterie (flot booléen)
sortie :	level	état de la minuterie (flot booléen)
constante :	delay	durée de la minuterie en secondes

Idée : réutiliser le nœud `stable`

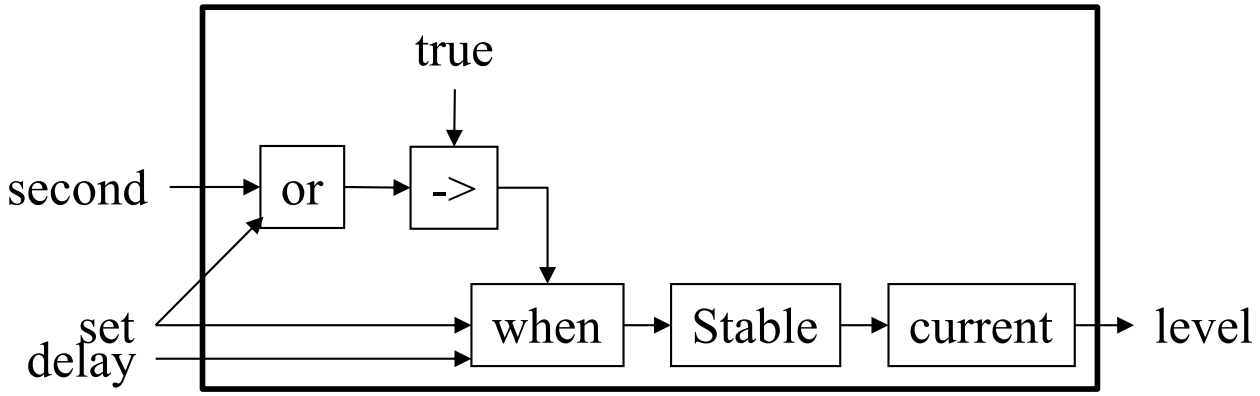
```
node stable_param (set, second : bool; delay : int)
returns (level : bool)
var ck : bool;
let
  level = current(stable((set, delay) when ck));
  ck = true -> (set or second);
tel.
```

=> Le nœud `stable` est appelé chaque fois que `ck` est vrai

=> L'horloge de l'instance de `stable` est `ck`

Lustre avec horloges : exemple

```
node stable_param (set, second : bool)
returns (level : bool)
var ck : bool;
let
  level = current(stable(set when ck));
  ck = true -> (set or second);
tel.
```



set	F	T	F	F	F	F	F	F	F	T
second	F	T	T	F	T	T	T	F	T	F
ck	T	T	T	F	T	T	T	F	T	T
count	0	5	4		3	2	1		0	5
level	F	T	T	T	T	T	T	T	F	T

Lustre avec horloges : à quoi ça sert ?

A contrôler les instants où un nœud est exécuté

Exemples

- Nœuds périodiques
- Nœuds soumis à des préconditions
- Nœuds exclusifs…

=> Voir BE2

Lustre : synthèse

Vérification sémantique :

Un programme Lustre est triplement vérifié

- vérification de types
- vérification de la causalité
  - Une et une seule équation par flot interne ou de sortie
  - Pas d'équation pour les flots d'entrée
  - Les dépendances de données (instantanées) entre équations forment un graphe acyclique
- **vérification d'horloge (vérification que le programme est bien « synchronisé »)**

⇒ Un programme Lustre qui compile est exécutable !

⇒ Et possibilité de vérifier formellement (parfois) le comportement d'un programme

2. Règles informelles du calcul d'horloge

Rappel syntaxique

Rappel syntaxique : un programme LUSTRE est composé

- d'une partie déclarative

```
X : type;
X : type when B ;
```

Soit *input*, *local*, *output* les listes des déclarations des flots d'entrée, locaux, de sortie.

- d'une partie équationnelle

```
Y = exp ;
```

```
exp ::= f(exp1,…, expn)          (avec f = +, -, *, /, or…)
      | exp1 when exp2
      | current(exp)
      | pre(exp)
      | exp1 -> exp2
      | X                        (flot)
      | val                      (valeur littérale ou constante)
```

Principes généraux du calcul d'horloge

Calcul d'horloge =

1. On calcule les fonctions suivantes

   - *clk_dec* : flot -> exp booléenne ∪ {`all`}

   fonction qui associe à tout flot son « horloge déclarée »

   - *clk_inf* : exp -> exp booléenne ∪ {`all`}

   fonction qui associe à toute expression de flot son « horloge inférée »

2. Puis on vérifie que *clk_dec* et *clk_inf* sont égales pour tous les flots du programme.

Règles informelle du calcul d'horloge

Horloge déclarée :

1. Tous les flots déclarés sans **when** ont comme horloge déclarée l'horloge de base du noeud

2. Tous les flots déclarés avec **when** ont comme horloge déclarée l'horloge déclarée par le **when**

Example:

```
node VOL100 (T:real; B:bool; P:real when B)
returns (V:real when B)
var T100:real when B;
```

clk_dec(T)= true

clk_dec(B)= true

clk_dec(P)= B

clk_dec(V)= B

clk_dec(T100)= B

Règles informelle du calcul d'horloge

Horloge inférée :

1. L'horloge inférée pour les flots d'entrée est leur horloge déclarée

2. Les constantes sont sur l'horloge de base

3. Soit une expression `f(exp1,…, expn)` avec f = `pre, ->, +, -, *, /, or`… (sans **`when`** ni **`current`**), alors
   - On ne peut inférer l'horloge de l'expression que si tous les exp_i ont la même horloge inférée
   - Et si c'est le cas, l'horloge inférée de l'expression est l'horloge inférée de chaque exp_i

4. Soit l'expression `X when B`, alors
   - On ne peut inférer l'horloge de l'expression que si X et B ont la même horloge inférée
   - Et si c'est le cas, l'horloge inférée de `X when B` est B

5. Soit l'expression **`current`** `X`, alors
   - On ne peut inférer l'horloge de l'expression que si l'horloge inférée de X n'est pas **`true`**
   - Et si c'est le cas, l'horloge inférée de **`current`** `X` est l'horloge inférée de l'horloge inférée de X

Règles informelle du calcul d'horloge

Horloge inférée :

Example:

```
const R : real;
node VOL100 (T:real; B:bool; P:real when B)
returns (V:real when B)
var T100:real when B;
let
  V= (T100 / P) * (R when B);
  T100 = T when B;
tel.
```

clk_dec(T)= true
clk_dec(B)= true
clk_dec(P)= B
clk_dec(V)= B
clk_dec(T100)= B

=>

clk_inf(T)= true
clk_inf(B)= true
clk_inf(P)= B

=>

clk_inf(T100)= B
clk_inf(P)= B

clk_inf(V)= B

Règles informelle du calcul d'horloge

Un programme est bien « synchronisé » (i.e., correct du point de vue des horloges) si :

- on est capable d'inférer une horloge pour tous les flots
- pour chaque flot, l'horloge déclarée et égale à l'horloge inférée