Programmation Concurrente, Réactive et Répartie

Cours N°7a

Carlos Agon & Emmanuel Chailloux

Master d'Informatique
Université Pierre et Marie Curie

année 2018-2019

Cours 7a : Langages synchrones à flots de données

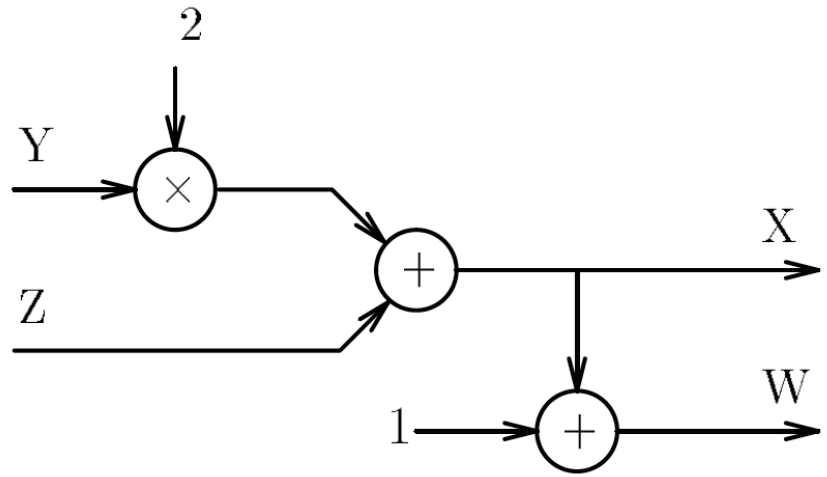
- programmes vus comme des équations
- causalité
- flots
- horloges
- exemples en Lustre et Esterel

Langages synchrones à flots de données

- Data Flow : un programme est donné par un ensemble d'équations et son exécution décrit l'évolution des sorties des équations dans le temps.
- Synchrony : un programme réagit à un événement externe dans un temps logique (suite d'instants) borné.
- Synchronous Data Flow : chaque instant de temps représente un cycle où les entrée sont prises en compte et les sorties calculées. Le temps est logique et ne peut pas être manipulé.

Programmes vus comme des équations

- un programme : un ensemble d'opérateurs reliés par des fils
- représentation graphique :



- représentation textuelle :

```
node Module1(Y,Z : int)
returns (X : int) ;
let
   X = (Y * 2) + Z ;
tel
```

```
node Module2 (X : int)
returns (W : int) ;
let
  W = X + 1 ;
tel
```

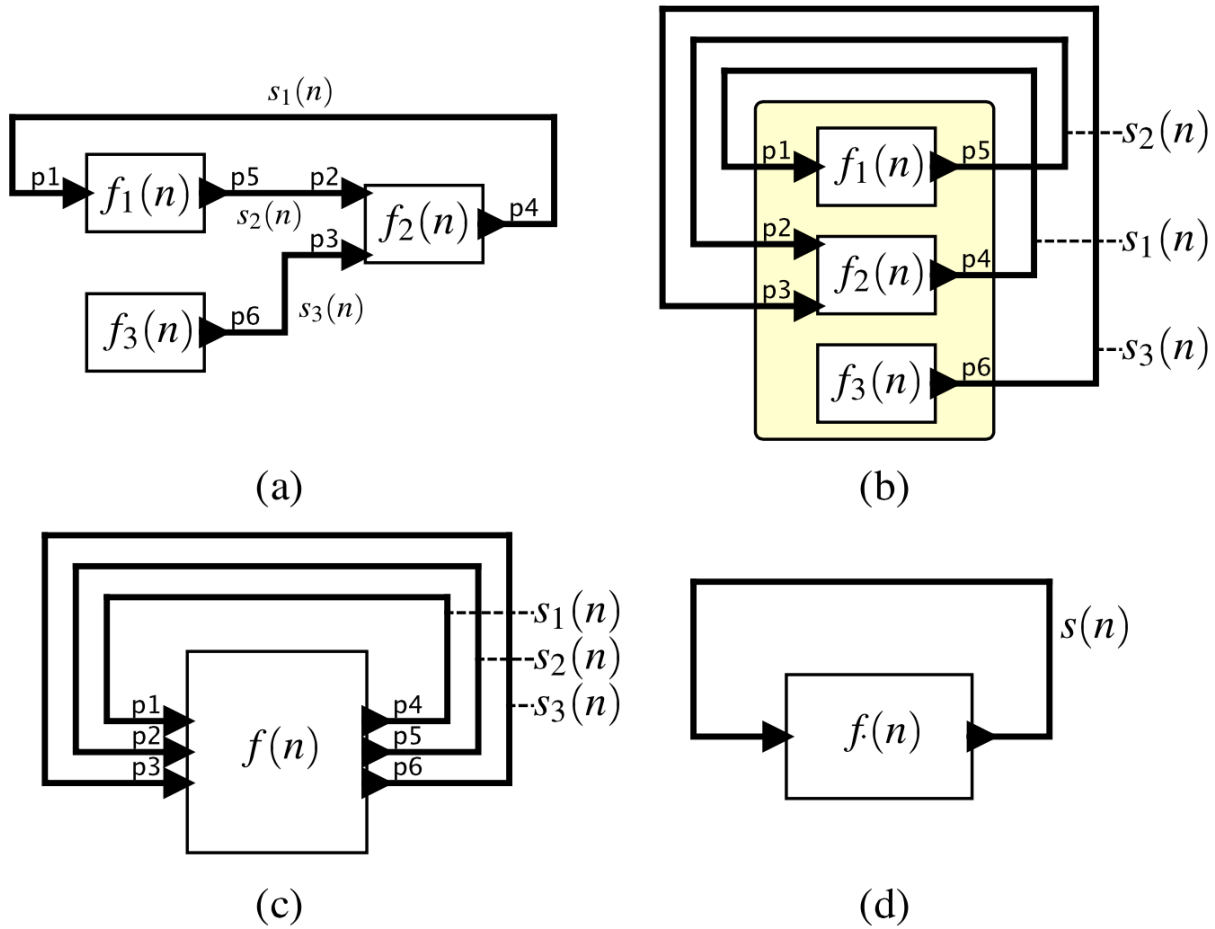
- représentation équationnelle : temps discret $= IN$

  $\forall t \in IN,\ X(t) = 2Y(t) + Z(t)$ et $W(t) = X(t) + 1$

```
node Module1(Y,Z : int)
returns (X : int) ;
var S : int ;              - variable locale
let
   X = S + Z ;             - equations
   S = Y * 2 ;             - sans ordre
tel
```

- toutes les variables sont des flots
- une seule équation pour chaque variable locale et de sortie
- ensemble d'équations à résoudre

Exécution d'un programme



Prefire (pré-conditions) + Fire (actions) + Postfire (post-conditions)
problème de causalité si les pré-conditions des actions sont influencées par les actions.

Langages

- LUSTRE : langage synchrone défini en 1985 par P. Caspi et N. Halbwachs (Vérimag, Grenoble)
  - vision fonctionnelle du monde
  - précision du contrôle par les horloges
  - capacité de conserver des valeurs dans des registres (`pre`) entre deux étapes de calcul
- SCADE : environnement de développement industriel développé par la société Esterel-Technologies
  - contient toujours un noyau Lustre mais a évolué
  - programmation graphique, génération de code C
  - utilisé dans le logiciel embqrqué critique (Airbus, ...)

LUSTRE : Types, opérateurs et conditionnelle

- affectation d'un flot de sortie : `X = ...`
- types de base : bool, int, real
- constante (flot constant)
- opérateurs arithmétiques, logiques, ...
- n-uplet
- if fonctionnel (expression) :

  `(bool flot) * ('a flot) * ('a flot) -> 'a flot`

```
node MinMax(A,B : real)
returns (R1,R2  : real) ;
let
  R1 = if (A <= B) then A else B ;
  R2 = if (A >= B) then A else B ;
tel
```

- parallélisme : eq_1 ; eq_2

flots LUSTRE (1)

$v_1 v_2 \ldots v_n \ldots$

- flot constant :
  - $2 \equiv 2,2,2,2,\ldots$
  - false $\equiv$ false, false, false, false, ...
- opérateurs :
  - $X \equiv x_0, x_1, x_2, x_3, \ldots$ et $Y \equiv y_0, y_1, y_2, y_3, \ldots$
    $X + Y \equiv x_0 + y_0, x_1 + y_1, x_2 + y_2, x_3 + y_3, \ldots$

flots LUSTRE (2)

Deux opérations temporelles :

- pre : un delay d'un pas (comme une case mémoire appelée registre).
- init : $x \rightarrow b_1..b_n..$: construit un nouveau stream $xb_1..b_n..$
- fby : $x \rightarrow preb$: construit un nouveau flot $xb_0b_1...b_n...$ que l'on peut lire par "followed by".

a	a1	a2	a3	a4 ...
b	b1	b2	b3	b4 ...
pre b	nil	b1	b2	b3 ...
a $\rightarrow$ b	a1	b2	b3	b4 ...
a $\rightarrow$ pre b	a1	b1	b2	b3 ...

Causalité en LUSTRE

LUSTRE n'autorise que les systèmes d'équations acycliques

- $x = \tau$ est acyclique, si x n'apparait pas dans τ ou seulement dans un sous-terme de type $pre(x)$ dans τ.
  - $a = a \; and \; pre(a)$ est cyclique
  - $a = b \; and \; pre(a)$ est acyclique
- propriété : les équations acycliques ont une unique solution

Horloges et échantillonage

Pour exprimer le contrôle en flot de données, on utilise les deux opérateurs suivants :

- opérateur `when` pour l'échantillonage
- opérateur `current` de projection

c	true	false	true	false	false	true ...
a	a1	a2	a3	a4	a5	a6 ...
b = a when c	a1		a3			a6 ...
c = current(b)	a1	a1	a3	a3	a3	a6

- c est appelé une horloge
- L'horloge de base est celle qui est toujours égale à true
- On ne peut pas faire `a + (a when c)`
- `current` ramène un flot sur une horloge plus rapide
- `current (a when c)` $\neq$ a

Combinaisons de flots échantillonnés (1)

c	true	false	true	false	false	true ...
a	a1	a2	a3	a4	a5	a6 ...
b	b1	b2	b3	b4	b5	b6 ...
a when c	a1		a3			a6 ...
b when c	b1		b3			b6 ...
(a when c) + (b when c)	a1 + b1		a3 + b3			a6 + b6 ...
pre (b when c)	nil		b1			b3 ...

- le dernier pre fonctionne par rapport à l'horloge c.
- Toute combinaison de flots est possible s'ils partagent la même horloge

Combinaisons de flots échantillonnés (2)

c	true	false	true	false	false	true ...
a	a1	a2	a3	a4	a5	a6 ...
b	b1	b2	b3	b4	b5	b6 ...
a when c	a1		a3			a6 ...
b when not c		b2		b4	b5	...
merge (c ; a when c ; b when not c)	a1	b2	a3	b4	b5	a6 ...

- Construction d'une horloge plus rapide
- Les deux flots doivent avoir des horloges complémentaires.

Exemple : ABRO (1)

émission d'un signal O dès qu'un signal A et un signal B ont été reçus, et répète ce processus à chaque fois qu'un signal R est reçu.

En Esterel :

```
module ABRO::
  input A, B, R;
  ouput O;

  loop
    [ await A || await B ] ;
    emit O
  each R

end module
```

Exemple : ABRO (2)

En Lustre : nodes EDGE (front montant) et ABRO

```
node EDGE(X : bool)
returns (Y : bool) ;
let
  Y = false -> X and not (pre X) ;
tel

node ABRO (A,B,R : bool)
returns (O : bool) ;
var seenA, seenB : bool ;
let
  O = EDGE(seenA and seenB) ;
  seenA = false ->  not R and (A or pre(seenA)) ;
  seenB = false ->  not R and (B or pre(seenB)) ;
tel
```

Exemple : Fibonnaci

```
node fibonacci() returns (f: int) ;
var x: int;
let
  f = 1->pre x ;
  x = 1 -> f + pre f ;
tel
```

f	1	1	2	3	5	8 ...
pre f	nil	1	1	2	3	5 ...

x	1	2	3	5	8	13 ...
pre x	nil	1	2	3	5	8 ...

```
node fibonacci() returns (f: int) ;
let
  f = 1->pre (1 -> f + pre f) ;
tel
```

Références

- supports au Collège de France (cours et séminaires de Gérard Berry)
- cours de Nicolas Halbwachs (Vérimag) et de Pascal Raymond (Vérimag)
- cours de Jean-Ferdinand Susini (Cnam)
- cours Esterel & Lustre d'Emmanuelle Encrenaz (UPMC)
- présentation de Scade par Jean-Louis Colaço (Esterel-Technologies)
- cours de Carlos Agon (Paradigmes de Programmation Concurrente - M2 STL)
- Lucid Syncrhone
  `https://www.di.ens.fr/~pouzet/lucid-synchrone/`
- OCaLustre :
  `https://www-apr.lip6.fr/~varoumas/docs/jfla.pdf`