

---

# TP 1 : A compiler for mini-Lustre

http:

`//www.di.ens.fr/~pouzet/cours/mpri/tp1/mini-lustre.tgz`

---

Systèmes Synchrones

Master Parisien de Recherche en Informatique

2019 – 2020

Marc Pouzet (et Louis Mandel)

# Normalisation

---

```
node f(x:int) returns (o:int);  
let  
  o = 1 fby 2 fby x;  
tel
```

```
node g() returns (o:int);  
let  
  o = f(f(1));  
tel
```

```
node f(x:int) returns (o:int);  
var aux1, aux2 : int;  
let  
  o = aux2;  
  aux2 = 1 fby aux1;  
  aux1 = 2 fby x;  
tel
```

```
node g() returns (o:int);  
var aux: int;  
let  
  o = f(aux);  
  aux = f(1);  
tel
```

# Static scheduling

---

```
node f(x:int) returns (o:int);  
var aux1, aux2 : int;  
let  
  o = aux2;  
  aux2 = 1 fby aux1;  
  aux1 = 2 fby x;  
tel
```

```
node g() returns (o:int);  
var aux: int;  
let  
  o = f(aux);  
  aux = f(1);  
tel
```

```
node f(x:int) returns (o:int);  
var aux1, aux2 : int;  
let  
  aux2 = 1 fby aux1;  
  aux1 = 2 fby x;  
  o = aux2;  
tel
```

```
node g() returns (o:int);  
var aux: int;  
let  
  aux = f(1);  
  o = f(aux);  
tel
```

# Imperative Code : compilation of fby

---

```
node f(x:int) returns (o:int);  
var aux1, aux2 : int;  
let  
  aux2 = 1 fby aux1;  
  aux1 = 2 fby x;  
  o = aux2;  
tel
```

```
type f_mem =  
  { mutable aux2_next: int;  
    mutable aux1_next: int; }  
let f_init () =  
  { aux2_next = 1;  
    aux1_next = 2; }  
let f_step mem x =  
  (* compute *)  
  let aux2 = mem.aux2_next in  
  let aux1 = mem.aux1_next in  
  let o = aux2 in  
  (* update *)  
  mem.aux2_next <- aux1;  
  mem.aux1_next <- x;  
  (* output *)  
  o
```

# Imperative Code : compilation of a function call

---

```
node g() returns (o:int);  
var aux: int;  
let  
    aux = f(1);  
    o = f(aux);  
tel
```

```
type g_mem =  
    { f_mem1: f_mem;  
      f_mem2: f_mem; }  
  
let g_init () =  
    { f_mem1 = f_init ();  
      f_mem2 = f_init (); }  
  
let g_step mem () =  
    (* compute *)  
    let aux = f_step mem.f_mem1 1 in  
    let o = f_step mem.f_mem2 aux in  
    (* update *)  
    ();  
    (* output *)  
    o
```

# Compilation of a program mini-Lustre

---

- ▶ `minilustre [options] file.mls`
  - ▷ `-main noeud` : simulation of node noeud
  - ▷ `-verbose` : print all the intermediate languages
- ▶ `ocamlc -thread unix.cma threads.cma graphics.cma file.ml`

<http://www.di.ens.fr/~pouzet/cours/mpri/tp1>