Une sémantique dénotationnelle pour un compilateur synchrone vérifié

Paul Jeanmaire

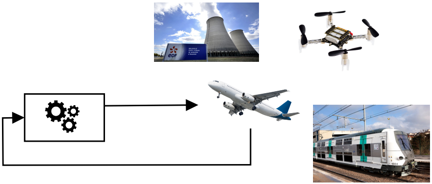
Thèse encadrée par Timothy Bourke et Marc Pouzet

à l'ENS, Inria, équipe Parkas

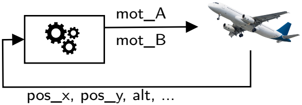
20 décembre 2024

Programmation des systèmes embarqués

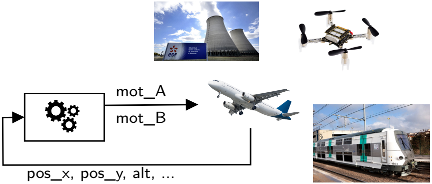
Programmation des systèmes embarqués



Programmation des systèmes embarqués

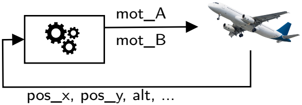


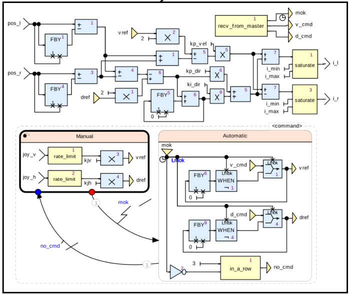
Programmation des systèmes embarqués




```
# Command line: ../countdown.c -c -S
        .text
        .align  16
        .globl fun$step$count_down
fun$step$count_down:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_adjust_cfa_offset  8
        leaq    16(%rsp), %rax
        movq    %rax, 0(%rsp)
        testl   %esi, %esi
        jne     .L100
        movzbl  0(%rdi), %esi
        testl   %esi, %esi
        jne     .L100
        movl    4(%rdi), %edx
.L100:
        xorl    %eax, %eax
        movb    %al, 0(%rdi)
        leal    -1(%edx), %ecx
        movl    %ecx, 4(%rdi)
        movq    %rdx, %rax
        addq    $8, %rsp
        ret
        .cfi_endproc
        .type   fun$step$count_down, @function
        .size   fun$step$count_down, . - fun$step$count_down
        .text
        .align  16
        .globl fun$reset$count_down
fun$reset$count_down:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_adjust_cfa_offset  8
        leaq    16(%rsp), %rax
        movq    %rax, 0(%rsp)
        movl    $1, %eax
        movb    %al, 0(%rdi)
        xorl    %ecx, %ecx
        movl    %ecx, 4(%rdi)
        addq    $8, %rsp
        ret
        .cfi_endproc
        .type   fun$reset$count_down, @function
        .size   fun$reset$count_down, . - fun$reset$count_down
```

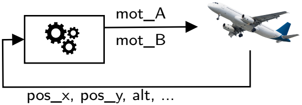
Programmation des systèmes embarqués

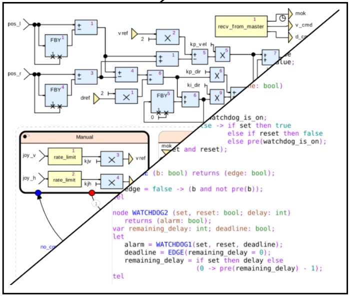





```
# Command line: ../countdown.c -c -S
        .text
        .align  16
        .globl fun$step$count_down
fun$step$count_down:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_adjust_cfa_offset  8
        leaq    16(%rsp), %rax
        movq    %rax, 0(%rsp)
        testl   %esi, %esi
        jne     .L100
        movzbl  0(%rdi), %esi
        testl   %esi, %esi
        jne     .L100
        movl    4(%rdi), %edx
.L100:
        xorl    %eax, %eax
        movb    %al, 0(%rdi)
        leal    -1(%edx), %ecx
        movl    %ecx, 4(%rdi)
        movq    %rdx, %rax
        addq    $8, %rsp
        ret
        .cfi_endproc
        .type   fun$step$count_down, @function
        .size   fun$step$count_down, . - fun$step$count_down
        .text
        .align  16
        .globl fun$reset$count_down
fun$reset$count_down:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_adjust_cfa_offset  8
        leaq    16(%rsp), %rax
        movq    %rax, 0(%rsp)
        movl    $1, %eax
        movb    %al, 0(%rdi)
        xorl    %ecx, %ecx
        movl    %ecx, 4(%rdi)
        addq    $8, %rsp
        ret
        .cfi_endproc
        .type   fun$reset$count_down, @function
        .size   fun$reset$count_down, . - fun$reset$count_down
```

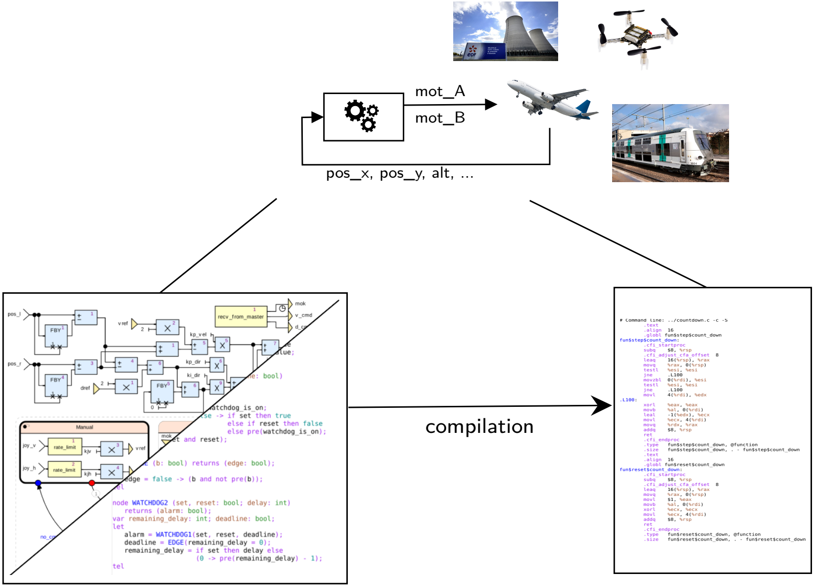
Programmation des systèmes embarqués



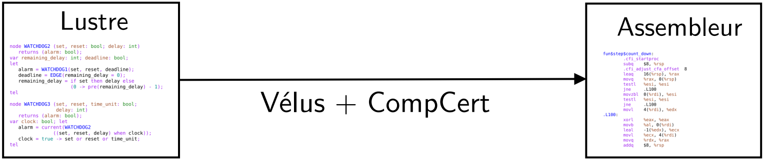



```
# Command line: ../countdown.c -c -S
        .text
        .align  16
        .globl fun$step$count_down
fun$step$count_down:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_adjust_cfa_offset  8
        leaq    16(%rsp), %rax
        movq    %rax, 0(%rsp)
        testl   %esi, %esi
        jne     .L100
        movzbl  0(%rdi), %esi
        testl   %esi, %esi
        jne     .L100
        movl    4(%rdi), %edx
.L100:
        xorl    %eax, %eax
        movb    %al, 0(%rdi)
        leal    -1(%edx), %ecx
        movl    %ecx, 4(%rdi)
        movq    %rdx, %rax
        addq    $8, %rsp
        ret
        .cfi_endproc
        .type   fun$step$count_down, @function
        .size   fun$step$count_down, . - fun$step$count_down
        .text
        .align  16
        .globl fun$reset$count_down
fun$reset$count_down:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_adjust_cfa_offset  8
        leaq    16(%rsp), %rax
        movq    %rax, 0(%rsp)
        movl    $1, %eax
        movb    %al, 0(%rdi)
        xorl    %ecx, %ecx
        movl    %ecx, 4(%rdi)
        addq    $8, %rsp
        ret
        .cfi_endproc
        .type   fun$reset$count_down, @function
        .size   fun$reset$count_down, . - fun$reset$count_down
```

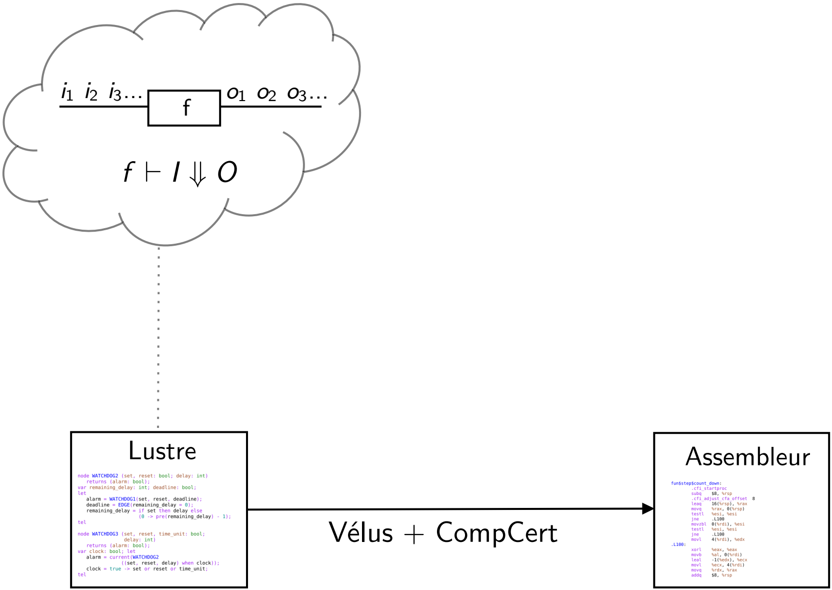
Programmation des systèmes embarqués



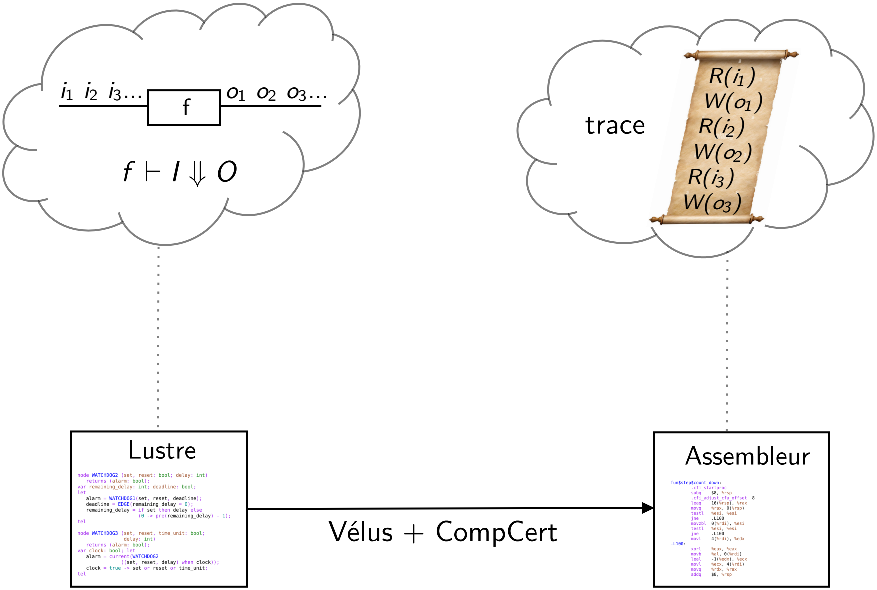
Compilation synchrone vérifiée



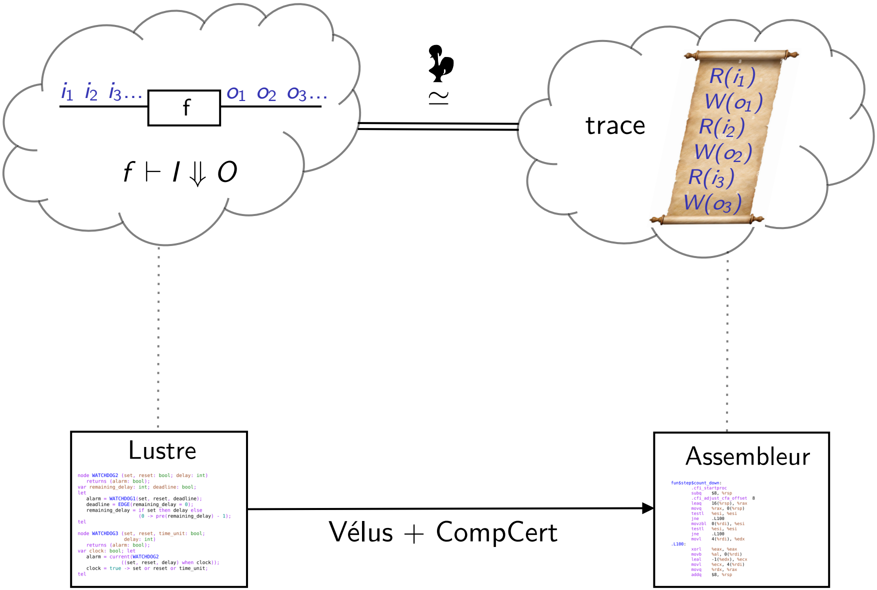
Compilation synchrone vérifiée



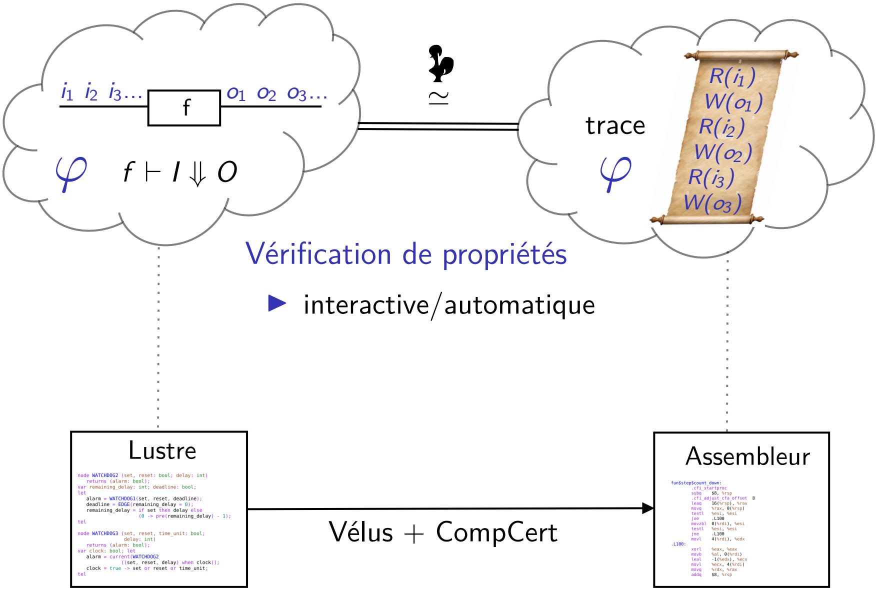
Compilation synchrone vérifiée



Compilation synchrone vérifiée



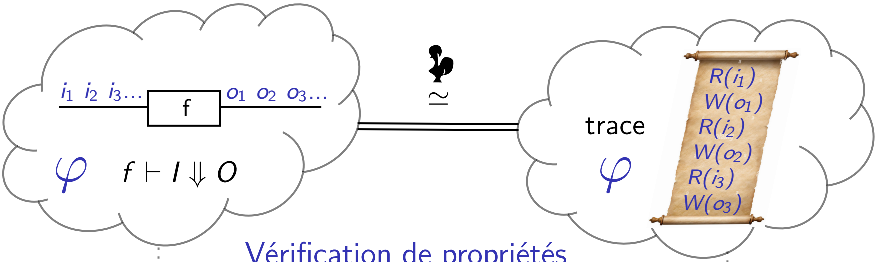
Compilation synchrone vérifiée



Vérification de propriétés

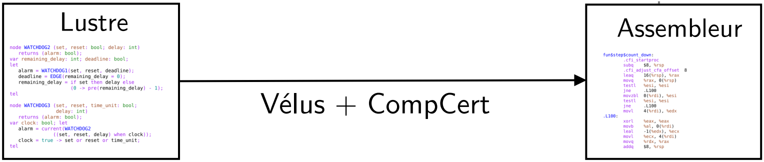
- interactive/automatique

Compilation synchrone vérifiée



Vérification de propriétés

- interactive/automatique
- lisibilité ?

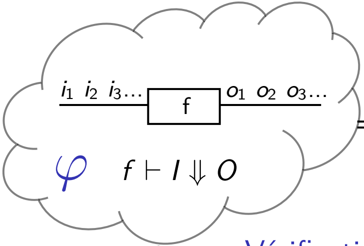


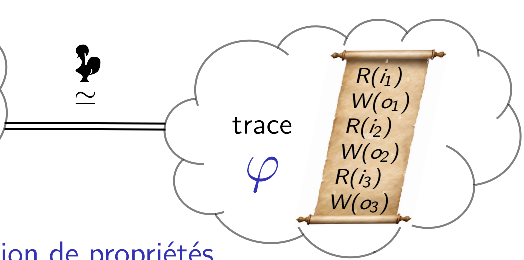
Compilation synchrone vérifiée

```
xs, ys, s1, s2, s3 : Stream svalue
H : History
bs := base_of [xs]
H1 : Env.find _r H = Some xs
H2 : Env.find _up H = Some s1
H3 : Env.find _pn H = Some s3
H4 : Env.find _n H = Some ys
H5 : fby (const bs (Venum 1)) s2 s1
H6 : sem_exp H bs ((_up and _n < 5) or (not _up and _n ≤ 1)) s2
H7 : sem_exp H bs (0 fby _n) s3
H8 : sem_exp H bs (if _up then (_pn + 1) else (_pn - 1)) ys
============================
Forall_Str (λ v ⇒ match v with
 | present (Vint i) ⇒
     Int.lt Int.zero i && Int.lt i (Int.repr 6) = true
 | _ ⇒ ⊥
 end) ys.
```

Vélus + CompCert

Compilation synchrone vérifiée





Vérification de propriétés

- interactive/automatique
- lisibilité ?

Lustre

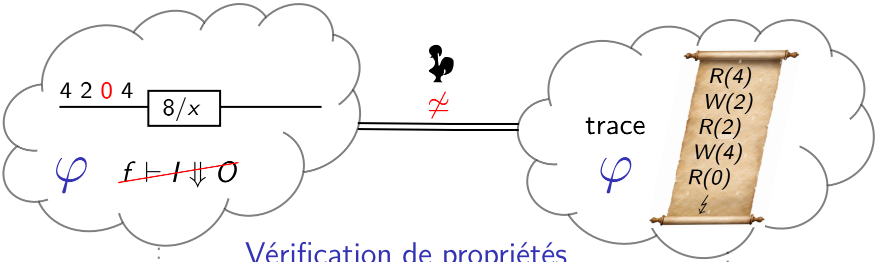
```
node WATCHDOG2 (set, reset: bool; delay: int)
   returns (alarm: bool);
var remaining_delay: int; deadline: bool;
let
   alarm = WATCHDOG1(set, reset, deadline);
   deadline = EDGE(remaining_delay = 0);
   remaining_delay = if set then delay else
                     (0 -> pre(remaining_delay) - 1);
tel

node WATCHDOG3 (set, reset, time_unit: bool;
                delay: int)
   returns (alarm: bool);
var clock: bool; let
   alarm = current(WATCHDOG2
               ((set, reset, delay) when clock));
   clock = true -> set or reset or time_unit;
tel
```

Vélus + CompCert

Assembleur

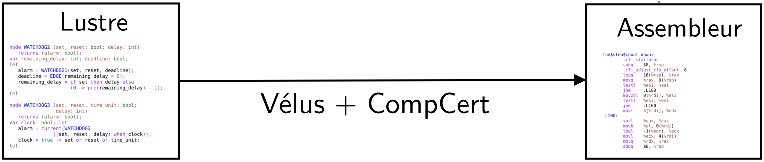
```
fun$step$count_down:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_adjust_cfa_offset  8
        leaq    16(%rsp), %rax
        movq    %rax, 0(%rsp)
        testl   %esi, %esi
        jne     .L100
        movzbl  0(%rdi), %esi
        testl   %esi, %esi
        jne     .L100
        movl    4(%rdi), %edx
.L100:
        xorl    %eax, %eax
        movb    %al, 0(%rdi)
        leal    -1(%edx), %ecx
        movl    %ecx, 4(%rdi)
        movq    %rdx, %rax
        addq    $8, %rsp
```

Compilation synchrone vérifiée

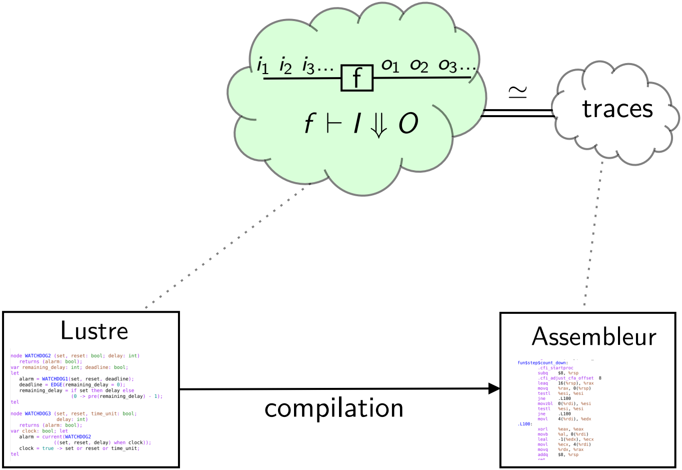


Vérification de propriétés

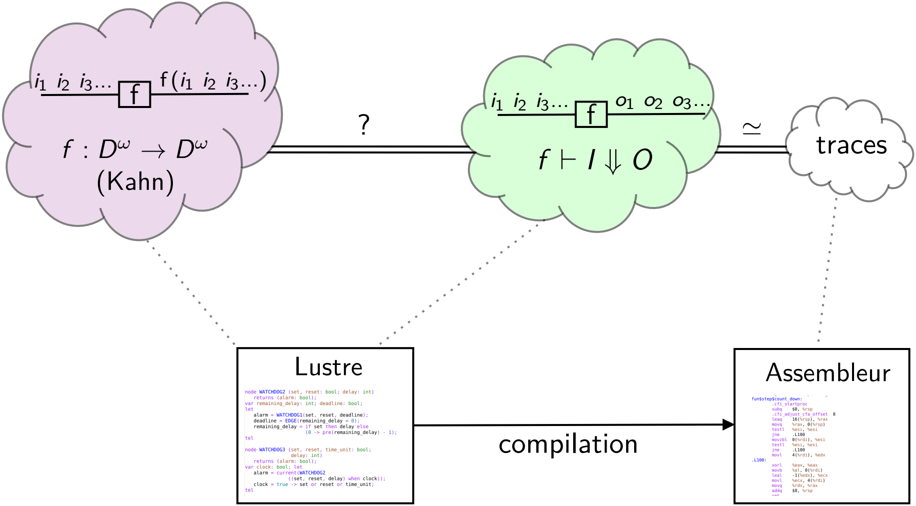
- interactive/automatique
- lisibilité ?
- traitement des erreurs ?



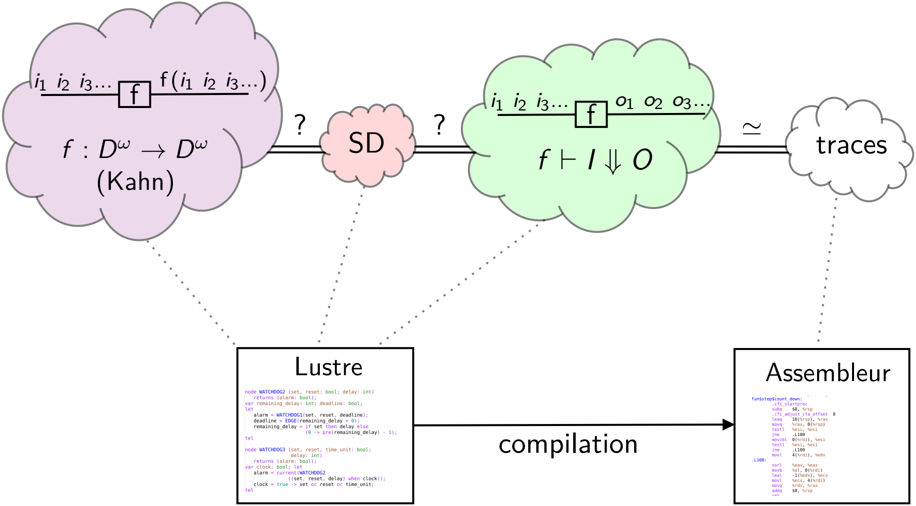
Approche proposée



Approche proposée



Approche proposée



Trois sémantiques à flots de données

	Kahn	Synchrone dénot.	Relationnelle
flots			
mécanique			
erreurs			

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots			D^∞ $D := \mathsf{A} \mid v$
mécanique			
erreurs			

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots	$D^\omega = D^* \cup D^\infty$ $D := v$		D^∞ $D := \mathsf{A} \mid v$
mécanique			
erreurs			

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := \mathsf{A} \mid v$	D^∞ $D := \mathsf{A} \mid v$
mécanique			
erreurs			

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := \mathsf{A} \mid v$	D^∞ $D := \mathsf{A} \mid v$
mécanique			relations/prédicats conjonctions existence supposée
erreurs			

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := \mathsf{A} \mid v$	D^∞ $D := \mathsf{A} \mid v$
mécanique	dénotationnelle (ordre préfixe), $\bot = \epsilon$ fonctions continues et bloquantes calcul du point fixe des équations		relations/prédicats conjonctions existence supposée
erreurs			

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := \mathsf{A} \mid v$	D^∞ $D := \mathsf{A} \mid v$
mécanique	dénotationnelle (ordre préfixe), $\bot = \epsilon$ fonctions continues et bloquantes calcul du point fixe des équations		relations/prédicats conjonctions existence supposée
erreurs			inexistantes

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := \mathsf{A} \mid v$	D^∞ $D := \mathsf{A} \mid v$
mécanique	dénotationnelle (ordre préfixe), $\bot = \epsilon$ fonctions continues et bloquantes calcul du point fixe des équations		relations/prédicats conjonctions existence supposée
erreurs	explicites (?)	explicites (?)	inexistantes

Trois sémantiques à flots de données

	Kahn	**Synchrone dénot.**	**Relationnelle**
flots	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := \mathsf{A} \mid v$	D^∞ $D := \mathsf{A} \mid v$
mécanique	dénotationnelle (ordre préfixe), $\bot = \epsilon$ fonctions continues et bloquantes calcul du point fixe des équations		relations/prédicats conjonctions existence supposée
erreurs	explicites (?)	explicites (?)	inexistantes

Problème d'encodage

- comment parler de D^ω en Coq ?
- par exemple, filter : $(D \to \mathbb{B}) \to D^\infty \to D^\omega$

Bibliothèque de flots

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

Domaines de Scott, (ω-)CPO, continuité, points fixes ...

Bibliothèque de flots

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

Domaines de Scott, (ω-)CPO, continuité, points fixes ...

$\text{fixp} : (D \to_c D) \to_c D$

$\text{fixp}\ F \simeq_D F(\text{fixp}\ F)$

$\forall F\ P,\ \text{admissible}\ P \to P\ \bot \to (\forall x,\ P\ x \to P(F\ x)) \to P(\text{fixp}\ F)$

Bibliothèque de flots

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

Domaines de Scott, (ω-)CPO, continuité, points fixes ...

$$\text{fixp} : (D \to_c D) \to_c D$$

$$\text{fixp } F \simeq_D F(\text{fixp } F)$$

$$\forall F\ P,\ \text{admissible } P \to P\ \bot \to (\forall x,\ P\ x \to P(F\ x)) \to P(\text{fixp } F)$$

Construction de D^ω

```
CoInductive Str (D : Type) :=
  | Cons : D -> Str D -> Str D
  | Tau : Str D -> Str D.

CoFixpoint bot := Tau bot.
```

Bibliothèque de flots

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

Domaines de Scott, (ω-)CPO, continuité, points fixes ...

$\text{fixp} : (D \to_c D) \to_c D$

$\text{fixp } F \simeq_D F(\text{fixp } F)$

$\forall F\ P,\ \text{admissible } P \to P\ \bot \to (\forall x,\ P\ x \to P(F\ x)) \to P(\text{fixp } F)$

Construction de D^ω

```
CoInductive Str (D : Type) :=
  | Cons : D -> Str D -> Str D
  | Tau : Str D -> Str D.

CoFixpoint bot := Tau bot.
```

$\bot \stackrel{\text{def}}{=} \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots$

Bibliothèque de flots

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

Domaines de Scott, (ω-)CPO, continuité, points fixes ...

$\mathrm{fixp} : (D \to_c D) \to_c D$

$\mathrm{fixp}\ F \simeq_D F(\mathrm{fixp}\ F)$

$\forall F\ P,\ \text{admissible}\ P \to P\ \bot \to (\forall x,\ P\ x \to P(F\ x)) \to P(\mathrm{fixp}\ F)$

Construction de D^ω

```
CoInductive Str (D : Type) :=
  | Cons : D -> Str D -> Str D
  | Tau : Str D -> Str D.

CoFixpoint bot := Tau bot.
```

$$\bot \stackrel{\text{def}}{=} \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots$$

$$[a; b] \stackrel{\text{def}}{=} a \cdot b \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots$$

Bibliothèque de flots

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

Domaines de Scott, (ω-)CPO, continuité, points fixes ...

$\mathrm{fixp} : (D \to_c D) \to_c D$

$\mathrm{fixp}\ F \simeq_D F(\mathrm{fixp}\ F)$

$\forall F\ P,\ \text{admissible}\ P \to P\ \bot \to (\forall x,\ P\ x \to P(F\ x)) \to P(\mathrm{fixp}\ F)$

Construction de D^ω

```
CoInductive Str (D : Type) :=
  | Cons : D -> Str D -> Str D
  | Tau : Str D -> Str D.

CoFixpoint bot := Tau bot.
```

$$\bot \stackrel{\text{def}}{=} \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots$$

$$[a; b] \stackrel{\text{def}}{=} a \cdot b \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots$$

$$\simeq \tau \cdot \tau \cdot a \cdot \tau \cdot b \cdot \tau \cdot \tau \cdots$$

Bibliothèque de flots

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

Domaines de Scott, (ω-)CPO, continuité, points fixes ...

$\text{fixp} : (D \to_c D) \to_c D$

$\text{fixp}\ F \simeq_D F(\text{fixp}\ F)$

$\forall F\ P,\ \text{admissible}\ P \to P\ \bot \to (\forall x,\ P\ x \to P(F\ x)) \to P(\text{fixp}\ F)$

Construction de D^ω

```
CoInductive Str (D : Type) :=
  | Cons : D -> Str D -> Str D
  | Tau : Str D -> Str D.

CoFixpoint bot := Tau bot.
```

$$\begin{aligned}
\bot &\stackrel{\text{def}}{=} \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots \\
[a; b] &\stackrel{\text{def}}{=} a \cdot b \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots \\
&\simeq \tau \cdot \tau \cdot a \cdot \tau \cdot b \cdot \tau \cdot \tau \cdots \\
&\preceq a \cdot b \cdot c \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots
\end{aligned}$$

Bibliothèque de flots (suite)

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

$$\text{filter} : (A \to \mathbb{B}) \to A^\omega \to_c A^\omega$$

$$\text{filter } f \ (a \cdot s) \simeq a \cdot \text{filter } f \ s \qquad \text{si } (f \ a) = \mathsf{T}$$

$$\text{filter } f \ (a \cdot s) \simeq \text{filter } f \ s \qquad \text{si } (f \ a) = \mathsf{F}$$

$$\text{filter } f \perp \simeq \perp$$

Bibliothèque de flots (suite)

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

$\mathrm{filter} : (A \to \mathbb{B}) \to A^\omega \to_c A^\omega$

$\mathrm{filter}\ f\ (a \cdot s) \simeq a \cdot \mathrm{filter}\ f\ s \qquad \text{si } (f\ a) = \mathsf{T}$

$\mathrm{filter}\ f\ (a \cdot s) \simeq \mathrm{filter}\ f\ s \qquad \text{si } (f\ a) = \mathsf{F}$

$\mathrm{filter}\ f\ \perp \simeq \perp$

$\mathrm{map} : (A \to B) \to A^\omega \to_c B^\omega$

$\mathrm{map}\ f\ (a \cdot s) \simeq (f\ a) \cdot \mathrm{map}\ f\ s$

$\mathrm{map}\ f\ \perp \simeq \perp$

Bibliothèque de flots (suite)

C. Paulin-Mohring, *A denotational semantics for Kahn networks in Coq*, 2009

$$\text{filter} : (A \to \mathbb{B}) \to A^\omega \to_c A^\omega$$

$$\text{filter } f\ (a \cdot s) \simeq a \cdot \text{filter } f\ s \qquad \text{si } (f\ a) = \mathsf{T}$$

$$\text{filter } f\ (a \cdot s) \simeq \text{filter } f\ s \qquad \text{si } (f\ a) = \mathsf{F}$$

$$\text{filter } f\ \bot \simeq \bot$$

$$\text{map} : (A \to B) \to A^\omega \to_c B^\omega$$

$$\text{map } f\ (a \cdot s) \simeq (f\ a) \cdot \text{map } f\ s$$

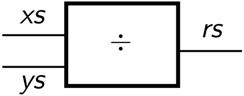
$$\text{map } f\ \bot \simeq \bot$$

$$\text{zip} : (A \to B \to C) \to A^\omega \to_c B^\omega \to_c C^\omega$$

$$\text{zip } f\ (a \cdot s_1)\ (b \cdot s_2) \simeq (f\ a\ b) \cdot \text{zip } f\ s_1\ s_2$$

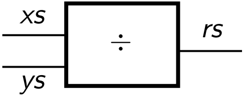
$$\text{zip } f\ s\ \bot \simeq \text{zip } f\ \bot\ s \simeq \bot$$

Construction : opération combinatoire



xs	A	10	10	A	10	10	⋯
ys	A	10	5	A	2	1	⋯
rs	A	1	2	A	5	10	⋯

Construction : opération combinatoire

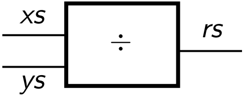


xs	A	10	10	A	10	10	$\cdots$
ys	A	10	5	A	2	1	$\cdots$
rs	A	1	2	A	5	10	$\cdots$

$$\frac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}$$

$$\frac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \mathsf{Some}\ r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}$$

Construction : opération combinatoire



xs	A	10	10	A	10	10	$\cdots$
ys	A	10	5	A	2	1	$\cdots$
rs	A	1	2	A	5	10	$\cdots$

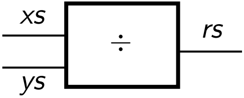
lift : $\mathsf{Str} \to_c \mathsf{Str} \to_c \mathsf{Str} :=$

zip

$$\frac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}$$

$$\frac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \mathsf{Some}\ r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}$$

Construction : opération combinatoire



xs	A	10	10	A	10	10	$\cdots$
ys	A	10	5	A	2	1	$\cdots$
rs	A	1	2	A	5	10	$\cdots$

$\mathsf{lift} \quad : \mathsf{Str} \to_c \mathsf{Str} \to_c \mathsf{Str} :=$

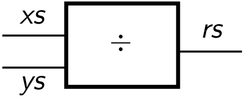
$\mathsf{zip}\,(\lambda\ \mathsf{A}, \mathsf{A} \to \mathsf{A}$

$)$

$$\frac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}$$

$$\frac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \mathsf{Some}\ r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}$$

Construction : opération combinatoire



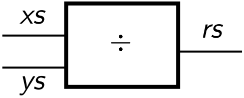
xs	A	10	10	A	10	10	$\cdots$
ys	A	10	5	A	2	1	$\cdots$
rs	A	1	2	A	5	10	$\cdots$

$$
\begin{array}{l}
\mathsf{lift} \quad : \mathsf{Str} \to_c \mathsf{Str} \to_c \mathsf{Str} := \\
\mathsf{zip}\,(\lambda\ \mathsf{A}, \mathsf{A} \to \mathsf{A} \\
\qquad \mid v_1, v_2 \to (\mathsf{match}\ v_1 \div v_2\ \mathsf{with} \\
\qquad\qquad\qquad \mid \mathsf{Some}\ r \to r \\
\qquad\qquad\qquad \mid \mathsf{None} \to \mathsf{err}_{\mathsf{rt}}) \\
\qquad\qquad\qquad\qquad)
\end{array}
$$

$$
\dfrac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}
$$

$$
\dfrac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \mathsf{Some}\ r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}
$$

Construction : opération combinatoire



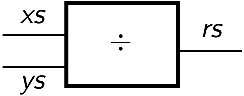
xs	A	10	10	A	10	10	$\cdots$
ys	A	10	5	A	2	1	$\cdots$
rs	A	1	2	A	5	10	$\cdots$

$$
\begin{array}{l}
\mathsf{lift} \quad : \mathsf{Str} \to_c \mathsf{Str} \to_c \mathsf{Str} := \\
\mathsf{zip}\,(\lambda\ \mathsf{A}, \mathsf{A} \to \mathsf{A} \\
\quad \mid v_1, v_2 \to (\mathsf{match}\ v_1 \div v_2\ \mathsf{with} \\
\qquad\qquad\quad \mid \mathsf{Some}\ r \to r \\
\qquad\qquad\quad \mid \mathsf{None} \to \mathsf{err}_{\mathsf{rt}}) \\
\quad \mid v, \mathsf{A} \mid \mathsf{A}, v \to \mathsf{err}_{\mathsf{sync}} \\
\qquad\qquad\qquad\qquad)
\end{array}
$$

$$
\frac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}
$$

$$
\frac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \mathsf{Some}\ r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}
$$

Construction : opération combinatoire



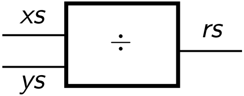
xs	A	10	10	A	10	10	$\cdots$
ys	A	10	5	A	2	1	$\cdots$
rs	A	1	2	A	5	10	$\cdots$

$$
\begin{array}{l}
\mathsf{lift} \quad : \mathsf{Str} \to_c \mathsf{Str} \to_c \mathsf{Str} := \\
\mathsf{zip}\,(\lambda\ \mathsf{A}, \mathsf{A} \to \mathsf{A} \\
\qquad \mid v_1, v_2 \to (\mathsf{match}\ v_1 \div v_2\ \mathsf{with} \\
\qquad\qquad\qquad\qquad \mid \mathsf{Some}\ r \to r \\
\qquad\qquad\qquad\qquad \mid \mathsf{None} \to \mathsf{err}_{\mathsf{rt}}) \\
\qquad \mid v, \mathsf{A} \mid \mathsf{A}, v \to \mathsf{err}_{\mathsf{sync}} \\
\qquad \mid \mathsf{err}, \_ \mid \_, \mathsf{err} \to \mathsf{err})
\end{array}
$$

$$
\frac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}
$$

$$
\frac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \mathsf{Some}\ r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}
$$

Construction : opération combinatoire



xs	A	10	10	A	10	10	⋯
ys	A	10	5	A	2	1	⋯
rs	A	1	2	A	5	10	⋯

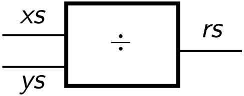
$$
\begin{array}{l}
\text{lift} \quad : \text{Str} \rightarrow_c \text{Str} \rightarrow_c \text{Str} := \\
\text{zip}\,(\lambda\ \text{A}, \text{A} \rightarrow \text{A} \\
\qquad \mid v_1, v_2 \rightarrow (\text{match } v_1 \div v_2 \text{ with} \\
\qquad\qquad\qquad\qquad \mid \text{Some } r \rightarrow r \\
\qquad\qquad\qquad\qquad \mid \text{None} \rightarrow \text{err}_{\text{rt}}) \\
\qquad \mid v, \text{A} \mid \text{A}, v \rightarrow \text{err}_{\text{sync}} \\
\qquad \mid \text{err}, \_ \mid \_, \text{err} \rightarrow \text{err})
\end{array}
$$

$$
\dfrac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\text{A} \cdot xs)\ (\text{A} \cdot ys)\ (\text{A} \cdot rs)}
$$

$$
\dfrac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \text{Some } r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}
$$

Théorème : si lift *xs ys* est sans erreurs, alors $\textsc{lift}\ xs\ ys\ (\text{lift}\ xs\ ys)$

Construction : opération combinatoire



xs	A	10	10	A	10	10	$\cdots$
ys	A	10	5	A	2	1	$\cdots$
rs	A	1	2	A	5	10	$\cdots$

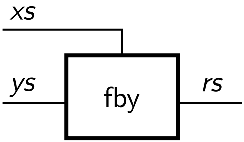
$$
\begin{array}{l}
\mathsf{lift}^{\#} : \mathsf{Str} \rightarrow_c \mathsf{Str} \rightarrow_c \mathsf{Str} := \\
\mathsf{zip}\,(\lambda\ \cancel{\mathsf{A}, \mathsf{A} \rightarrow \mathsf{A}} \\
\quad \mid v_1, v_2 \rightarrow (\mathsf{match}\ v_1 \div v_2\ \mathsf{with} \\
\qquad\qquad\qquad \mid \mathsf{Some}\ r \rightarrow r \\
\qquad\qquad\qquad \mid \mathsf{None} \rightarrow \mathsf{err}_{\mathsf{rt}}) \\
\quad \mid \cancel{v, \mathsf{A} \mid \mathsf{A}, v \rightarrow \mathsf{err}_{\mathsf{sync}}} \\
\quad \mid \mathsf{err}, \_ \mid \_, \mathsf{err} \rightarrow \mathsf{err})
\end{array}
$$

$$
\frac{\textsc{lift}\ xs\ ys\ rs}{\textsc{lift}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}
$$

$$
\frac{\textsc{lift}\ xs\ ys\ rs \qquad v_1 \div v_2 = \mathsf{Some}\ r}{\textsc{lift}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (r \cdot rs)}
$$

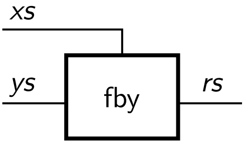
- **Théorème** : si lift xs ys est sans erreurs, alors $\textsc{lift}\ xs\ ys\ (\mathsf{lift}\ xs\ ys)$
- **Théorème** : et aussi, $\lfloor \mathsf{lift}\ xs\ ys \rfloor_{\mathsf{A}} \simeq \mathsf{lift}^{\#}\ \lfloor xs \rfloor_{\mathsf{A}}\ \lfloor ys \rfloor_{\mathsf{A}}$

Construction : délai initialisé



xs	5	5	5	5	5	5	$\cdots$
ys	6	7	8	9	10	11	$\cdots$
rs	5	6	7	8	9	10	$\cdots$

Construction : délai initialisé



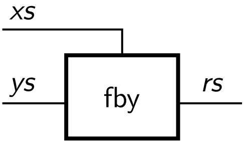
xs	5	5	5	5	5	5	$\cdots$
ys	6	7	8	9	10	11	$\cdots$
rs	5	6	7	8	9	10	$\cdots$

$$\mathrm{fby}^{\#} : \mathrm{Str} \to_c \mathrm{Str} \to_c \mathrm{Str}$$

$$\mathrm{fby}^{\#}\ \bot\ ys \simeq \bot$$

$$\mathrm{fby}^{\#}\ (v \cdot xs)\ ys \simeq v \cdot ys$$

Construction : délai initialisé



xs	A	5	A	5	5	5	5	$\cdots$
ys	A	6	A	7	8	9	10	$\cdots$
rs	A	5	A	6	7	8	9	$\cdots$

$$\mathsf{fby}^{\#} : \mathsf{Str} \to_c \mathsf{Str} \to_c \mathsf{Str}$$

$$\mathsf{fby}^{\#}\ \bot\ ys \simeq \bot$$

$$\mathsf{fby}^{\#}\ (v \cdot xs)\ ys \simeq v \cdot ys$$

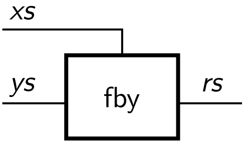
$$\frac{\text{FBY}\ xs\ ys\ rs}{\text{FBY}\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}$$

$$\frac{\text{FBY}_1\ v_2\ xs\ ys\ rs}{\text{FBY}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (v_1 \cdot rs)}$$

$$\frac{\text{FBY}_1\ v\ xs\ ys\ rs}{\text{FBY}_1\ v\ (\mathsf{A} \cdot xs)\ (\mathsf{A} \cdot ys)\ (\mathsf{A} \cdot rs)}$$

$$\frac{\text{FBY}_1\ v_2\ xs\ ys\ rs}{\text{FBY}_1\ v\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (v \cdot rs)}$$

Construction : délai initialisé



xs	A	5	A	5	5	5	5	$\cdots$
ys	A	6	A	7	8	9	10	$\cdots$
rs	A	5	A	6	7	8	9	$\cdots$

$$\mathrm{fby}^\# : \mathrm{Str} \to_c \mathrm{Str} \to_c \mathrm{Str}$$

$$\mathrm{fby}^\# \perp ys \simeq \perp$$

$$\mathrm{fby}^\# (v \cdot xs)\ ys \simeq v \cdot ys$$

$$\frac{\mathrm{FBY}\ xs\ ys\ rs}{\mathrm{FBY}\ (\mathrm{A} \cdot xs)\ (\mathrm{A} \cdot ys)\ (\mathrm{A} \cdot rs)}$$

$$\frac{\mathrm{FBY}_1\ v_2\ xs\ ys\ rs}{\mathrm{FBY}\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (v_1 \cdot rs)}$$

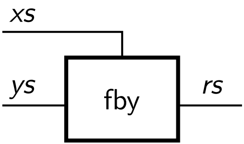
$$\frac{\mathrm{FBY}_1\ v\ xs\ ys\ rs}{\mathrm{FBY}_1\ v\ (\mathrm{A} \cdot xs)\ (\mathrm{A} \cdot ys)\ (\mathrm{A} \cdot rs)}$$

$$\frac{\mathrm{FBY}_1\ v_2\ xs\ ys\ rs}{\mathrm{FBY}_1\ v\ (v_1 \cdot xs)\ (v_2 \cdot ys)\ (v \cdot rs)}$$

Synchrone dénotationnel

- quelles erreurs ?
- $x = 0$ fby $(x + 1)$?

Construction : délai initialisé



xs	A	5	A	5	5	5	5	$\cdots$
ys	A	6	A	7	8	9	10	$\cdots$
rs	A	5	A	6	7	8	9	$\cdots$

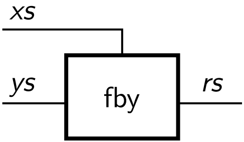
$$\mathrm{fby}\ (\mathrm{A} \cdot xs)\ ys := \mathrm{A} \cdot \mathrm{fby}_{\mathrm{A}}\ xs\ ys$$
$$\mathrm{fby}\ (v \cdot xs)\ ys := v \cdot \mathrm{fby}'_1\ \mathrm{None}\ xs\ ys$$

$$\mathrm{fby}'_1\ \mathrm{None}\ xs\ (v \cdot ys) := \mathrm{fby}_1\ v\ xs\ ys$$

$$\mathrm{fby}_{\mathrm{A}}\ xs\ (\mathrm{A} \cdot ys) := \mathrm{fby}\ xs\ ys$$

$$\mathrm{fby}_1\ v\ (\mathrm{A} \cdot xs)\ ys := \mathrm{A} \cdot \mathrm{fby}'_1\ (\mathrm{Some}\ v)\ xs\ ys$$
$$\mathrm{fby}_1\ v\ (v_x \cdot xs)\ ys := v \cdot \mathrm{fby}'_1\ \mathrm{None}\ xs\ ys$$

Construction : délai initialisé



xs	A	5	A	5	5	5	5	$\cdots$
ys	A	6	A	7	8	9	10	$\cdots$
rs	A	5	A	6	7	8	9	$\cdots$

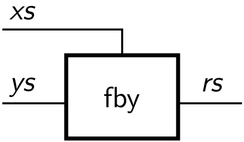
$$\begin{aligned}
\mathrm{fby}\ (\mathrm{A} \cdot xs)\ ys &:= \mathrm{A} \cdot \mathrm{fby}_{\mathrm{A}}\ xs\ ys \\
\mathrm{fby}\ (v \cdot xs)\ ys &:= v \cdot \mathrm{fby}'_1\ \mathrm{None}\ xs\ ys \\
\mathrm{fby}\ (\mathrm{err} \cdot xs)\ ys &:= \mathrm{err} \cdot \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs
\end{aligned}$$

$$\begin{aligned}
\mathrm{fby}_{\mathrm{A}}\ xs\ (\mathrm{A} \cdot ys) &:= \mathrm{fby}\ xs\ ys \\
\mathrm{fby}_{\mathrm{A}}\ xs\ (\mathrm{err} \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs \\
\mathrm{fby}_{\mathrm{A}}\ xs\ (v \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err}_{\mathrm{sync}})\ xs
\end{aligned}$$

$$\begin{aligned}
\mathrm{fby}'_1\ \mathrm{None}\ xs\ (v \cdot ys) &:= \mathrm{fby}_1\ v\ xs\ ys \\
\mathrm{fby}'_1\ (\mathrm{Some}\ v)\ xs\ (\mathrm{A} \cdot ys) &:= \mathrm{fby}_1\ v\ xs\ ys \\
\mathrm{fby}'_1\ _\ xs\ (\mathrm{err} \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs \\
\mathrm{fby}'_1\ _\ xs\ (_ \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err}_{\mathrm{sync}})\ xs
\end{aligned}$$

$$\begin{aligned}
\mathrm{fby}_1\ v\ (\mathrm{A} \cdot xs)\ ys &:= \mathrm{A} \cdot \mathrm{fby}'_1\ (\mathrm{Some}\ v)\ xs\ ys \\
\mathrm{fby}_1\ v\ (v_x \cdot xs)\ ys &:= v \cdot \mathrm{fby}'_1\ \mathrm{None}\ xs\ ys \\
\mathrm{fby}_1\ v\ (\mathrm{err} \cdot xs)\ ys &:= \mathrm{err} \cdot \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs
\end{aligned}$$

Construction : délai initialisé



xs	A	5	A	5	5	5	5	⋯
ys	A	6	A	7	8	9	10	⋯
rs	A	5	A	6	7	8	9	⋯

$$\begin{aligned}
\mathsf{fby}\ (\mathsf{A}\cdot xs)\ ys &:= \mathsf{A}\cdot \mathsf{fby}_{\mathsf{A}}\ xs\ ys\\
\mathsf{fby}\ (v\cdot xs)\ ys &:= v\cdot \mathsf{fby}'_1\ \mathsf{None}\ xs\ ys\\
\mathsf{fby}\ (\mathsf{err}\cdot xs)\ ys &:= \mathsf{err}\cdot \mathsf{map}\ (\lambda x.\ \mathsf{err})\ xs
\end{aligned}$$

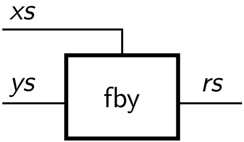
$$\begin{aligned}
\mathsf{fby}_{\mathsf{A}}\ xs\ (\mathsf{A}\cdot ys) &:= \mathsf{fby}\ xs\ ys\\
\mathsf{fby}_{\mathsf{A}}\ xs\ (\mathsf{err}\cdot ys) &:= \mathsf{map}\ (\lambda x.\ \mathsf{err})\ xs\\
\mathsf{fby}_{\mathsf{A}}\ xs\ (v\cdot ys) &:= \mathsf{map}\ (\lambda x.\ \mathsf{err}_{\mathsf{sync}})\ xs
\end{aligned}$$

$$\begin{aligned}
\mathsf{fby}'_1\ \mathsf{None}\ xs\ (v\cdot ys) &:= \mathsf{fby}_1\ v\ xs\ ys\\
\mathsf{fby}'_1\ (\mathsf{Some}\ v)\ xs\ (\mathsf{A}\cdot ys) &:= \mathsf{fby}_1\ v\ xs\ ys\\
\mathsf{fby}'_1\ _\ xs\ (\mathsf{err}\cdot ys) &:= \mathsf{map}\ (\lambda x.\ \mathsf{err})\ xs\\
\mathsf{fby}'_1\ _\ xs\ (_\cdot ys) &:= \mathsf{map}\ (\lambda x.\ \mathsf{err}_{\mathsf{sync}})\ xs
\end{aligned}$$

$$\begin{aligned}
\mathsf{fby}_1\ v\ (\mathsf{A}\cdot xs)\ ys &:= \mathsf{A}\cdot \mathsf{fby}'_1\ (\mathsf{Some}\ v)\ xs\ ys\\
\mathsf{fby}_1\ v\ (v_x\cdot xs)\ ys &:= v\cdot \mathsf{fby}'_1\ \mathsf{None}\ xs\ ys\\
\mathsf{fby}_1\ v\ (\mathsf{err}\cdot xs)\ ys &:= \mathsf{err}\cdot \mathsf{map}\ (\lambda x.\ \mathsf{err})\ xs
\end{aligned}$$

Théorème : si fby *xs ys* est sans erreurs, alors $\mathrm{FBY}\ xs\ ys\ (\mathsf{fby}\ xs\ ys)$

Construction : délai initialisé



xs	A	5	A	5	5	5	5	$\cdots$
ys	A	6	A	7	8	9	10	$\cdots$
rs	A	5	A	6	7	8	9	$\cdots$

$$
\begin{aligned}
\mathrm{fby}\ (\mathsf{A} \cdot xs)\ ys &:= \mathsf{A} \cdot \mathrm{fby}_{\mathsf{A}}\ xs\ ys \\
\mathrm{fby}\ (v \cdot xs)\ ys &:= v \cdot \mathrm{fby}'_1\ \mathrm{None}\ xs\ ys \\
\mathrm{fby}\ (\mathrm{err} \cdot xs)\ ys &:= \mathrm{err} \cdot \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs
\end{aligned}
$$

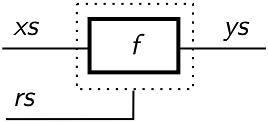
$$
\begin{aligned}
\mathrm{fby}_{\mathsf{A}}\ xs\ (\mathsf{A} \cdot ys) &:= \mathrm{fby}\ xs\ ys \\
\mathrm{fby}_{\mathsf{A}}\ xs\ (\mathrm{err} \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs \\
\mathrm{fby}_{\mathsf{A}}\ xs\ (v \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err}_{\mathrm{sync}})\ xs
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{fby}'_1\ \mathrm{None}\ xs\ (v \cdot ys) &:= \mathrm{fby}_1\ v\ xs\ ys \\
\mathrm{fby}'_1\ (\mathrm{Some}\ v)\ xs\ (\mathsf{A} \cdot ys) &:= \mathrm{fby}_1\ v\ xs\ ys \\
\mathrm{fby}'_1\ \_\ xs\ (\mathrm{err} \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs \\
\mathrm{fby}'_1\ \_\ xs\ (\_ \cdot ys) &:= \mathrm{map}\ (\lambda x.\ \mathrm{err}_{\mathrm{sync}})\ xs
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{fby}_1\ v\ (\mathsf{A} \cdot xs)\ ys &:= \mathsf{A} \cdot \mathrm{fby}'_1\ (\mathrm{Some}\ v)\ xs\ ys \\
\mathrm{fby}_1\ v\ (v_x \cdot xs)\ ys &:= v \cdot \mathrm{fby}'_1\ \mathrm{None}\ xs\ ys \\
\mathrm{fby}_1\ v\ (\mathrm{err} \cdot xs)\ ys &:= \mathrm{err} \cdot \mathrm{map}\ (\lambda x.\ \mathrm{err})\ xs
\end{aligned}
$$

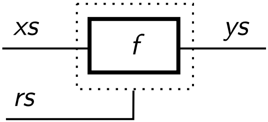
- Théorème : si fby $xs\ ys$ est sans erreurs, alors $\mathrm{FBY}\ xs\ ys\ (\mathrm{fby}\ xs\ ys)$
- Théorème : et aussi, $\lfloor \mathrm{fby}\ xs\ ys \rfloor_{\mathsf{A}} \preceq \mathrm{fby}^{\#}\ \lfloor xs \rfloor_{\mathsf{A}}\ \lfloor ys \rfloor_{\mathsf{A}}$

Construction : réinitialisation modulaire



xs	1	1	1	A	1	1	1	1	$\cdots$
rs	F	F	F	A	T	F	F	T	$\cdots$
ys	1	2	3	A	1	2	3	1	$\cdots$

Construction : réinitialisation modulaire

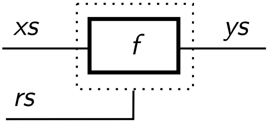


xs	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\cdots$
rs	F	F	F	A	T	F	F	T	$\cdots$
ys	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	$\cdots$

[Bourke, Brun, Pouzet, 2020]

$$
\begin{aligned}
& f \vdash [x_1 \cdot x_2 \cdot x_3 \cdot x_4] \Downarrow [y_1 \cdot y_2 \cdot y_3 \cdot y_4] \\
\wedge\ & f \vdash [x_5 \cdot x_6 \cdot x_7] \Downarrow [y_5 \cdot y_6 \cdot y_7] \\
\wedge\ & f \vdash [x_8] \Downarrow [y_8] \\
\wedge\ & \cdots
\end{aligned}
$$

Construction : réinitialisation modulaire



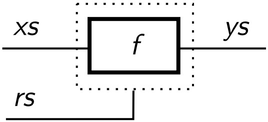
xs	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\cdots$
rs	F	F	F	A	T	F	F	T	$\cdots$
ys	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	$\cdots$

[Bourke, Brun, Pouzet, 2020]

$$
\begin{aligned}
& f \vdash [x_1 \cdot x_2 \cdot x_3 \cdot x_4] \Downarrow [y_1 \cdot y_2 \cdot y_3 \cdot y_4] \\
\wedge\ & f \vdash [x_5 \cdot x_6 \cdot x_7] \Downarrow [y_5 \cdot y_6 \cdot y_7] \\
\wedge\ & f \vdash [x_8] \Downarrow [y_8] \\
\wedge\ & \cdots
\end{aligned}
$$

(masquage : $[x_2 \cdot x_3] \equiv \mathsf{A} \cdot x_2 \cdot x_3 \cdot \mathsf{A} \cdot \mathsf{A} \cdots$)

Construction : réinitialisation modulaire



xs	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\cdots$
rs	F	F	F	A	T	F	F	T	$\cdots$
ys	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	$\cdots$

[Hamon, Pouzet, 2001]

$\text{reset}_f\ rs\ xs :=$

let $cs = \text{true-until}\ rs$ **in**

$\text{merge}\ cs\ (f\ (\text{when}\ cs\ xs))$

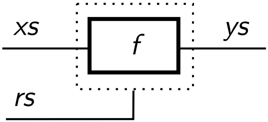
$\quad (\text{reset}_f\ (\text{whenot}\ cs\ rs)\ (\text{whenot}\ cs\ xs))$

[Bourke, Brun, Pouzet, 2020]

$$f \vdash [x_1 \cdot x_2 \cdot x_3 \cdot x_4] \Downarrow [y_1 \cdot y_2 \cdot y_3 \cdot y_4]$$
$$\wedge\ f \vdash [x_5 \cdot x_6 \cdot x_7] \Downarrow [y_5 \cdot y_6 \cdot y_7]$$
$$\wedge\ f \vdash [x_8] \Downarrow [y_8]$$
$$\wedge\ \cdots$$

(masquage : $[x_2 \cdot x_3] \equiv \text{A} \cdot x_2 \cdot x_3 \cdot \text{A} \cdot \text{A} \cdots$)

Construction : réinitialisation modulaire



xs	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\cdots$
rs	F	F	F	A	T	F	F	T	$\cdots$
ys	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	$\cdots$

[Hamon, Pouzet, 2001]

$\mathrm{reset}_f\ rs\ xs :=$

$\quad$ **let** $cs = \text{true-until}\ rs$ **in**

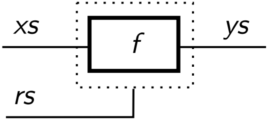
$\quad \mathrm{merge}\ cs\ (f\ (\mathrm{when}\ cs\ xs))$

$\qquad\qquad (\mathrm{reset}_f\ (\mathrm{whenot}\ cs\ rs)\ (\mathrm{whenot}\ cs\ xs))$

[Gérard, 2013]

$$
\begin{aligned}
\mathrm{sreset}_f\ rs\ xs &:= \mathrm{sreset}'_f\ rs\ xs\ (f\ xs)\\
\mathrm{sreset}'_f\ (T \cdot rs)\ xs\ ys &\simeq \mathrm{sreset}'_f\ (F \cdot rs)\ xs\ (f\ xs)\\
\mathrm{sreset}'_f\ (F \cdot rs)\ (x \cdot xs)\ (y \cdot ys) &\simeq y \cdot (\mathrm{sreset}'_f\ rs\ xs\ ys)\\
\mathrm{sreset}'_f\ (\mathrm{A} \cdot rs)\ (x \cdot xs)\ (y \cdot ys) &\simeq y \cdot (\mathrm{sreset}'_f\ rs\ xs\ ys)
\end{aligned}
$$

Construction : réinitialisation modulaire



xs	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\cdots$
rs	F	F	F	A	T	F	F	T	$\cdots$
ys	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	$\cdots$

[Hamon, Pouzet, 2001]

$\mathrm{reset}_f\ rs\ xs :=$
let $cs = \text{true-until}\ rs$ **in**
$\mathrm{merge}\ cs\ (f\ (\mathrm{when}\ cs\ xs))$
$\quad (\mathrm{reset}_f\ (\mathrm{whenot}\ cs\ rs)\ (\mathrm{whenot}\ cs\ xs))$

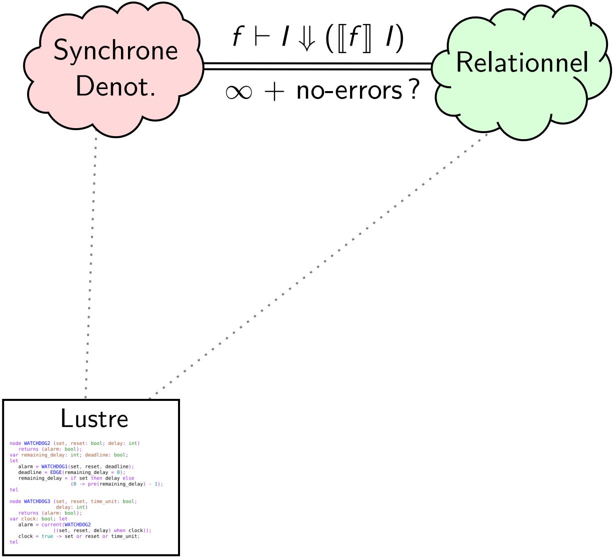
[Gérard, 2013]

$$
\begin{aligned}
\mathrm{sreset}_f\ rs\ xs &:= \mathrm{sreset}'_f\ rs\ xs\ (f\ xs)\\
\mathrm{sreset}'_f\ (T \cdot rs)\ xs\ ys &\simeq \mathrm{sreset}'_f\ (F \cdot rs)\ xs\ (f\ xs)\\
\mathrm{sreset}'_f\ (F \cdot rs)\ (x \cdot xs)\ (y \cdot ys) &\simeq y \cdot (\mathrm{sreset}'_f\ rs\ xs\ ys)\\
\mathrm{sreset}'_f\ (A \cdot rs)\ (x \cdot xs)\ (y \cdot ys) &\simeq y \cdot (\mathrm{sreset}'_f\ rs\ xs\ ys)
\end{aligned}
$$

Théorème : si $\mathrm{clock}(rs) \simeq \mathrm{clock}(xs)$ alors $\mathrm{reset}_f\ rs\ xs \simeq \mathrm{sreset}_f\ rs\ xs$

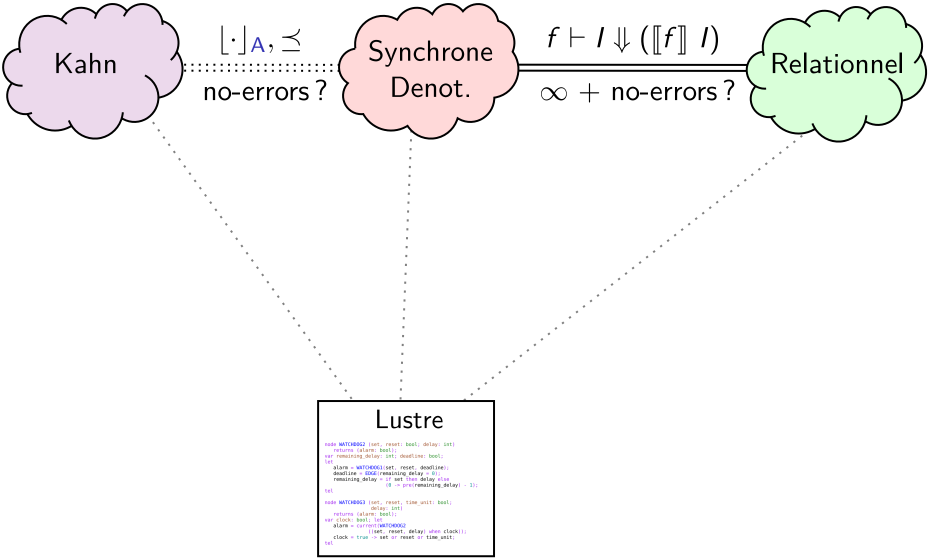
Approche proposée

Traitement des erreurs



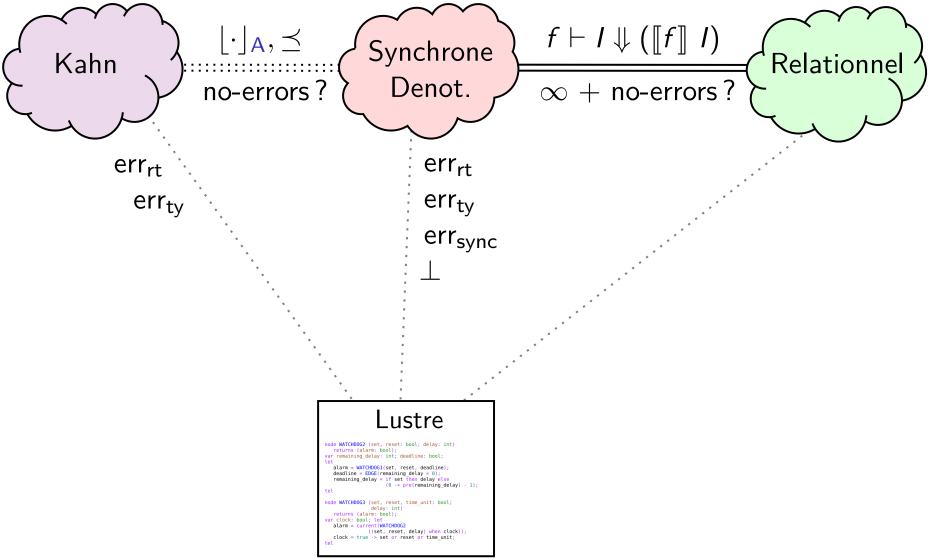
Approche proposée

Traitement des erreurs



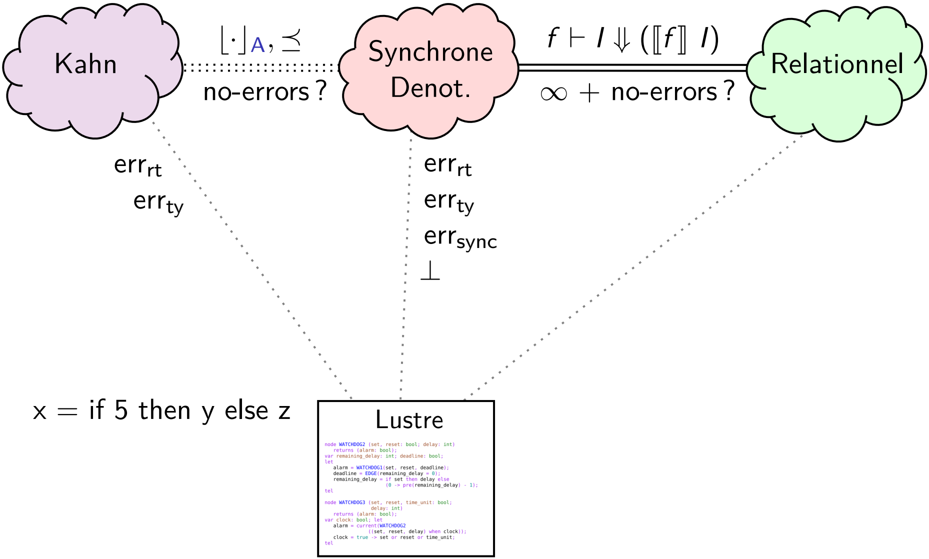
Approche proposée

Traitement des erreurs



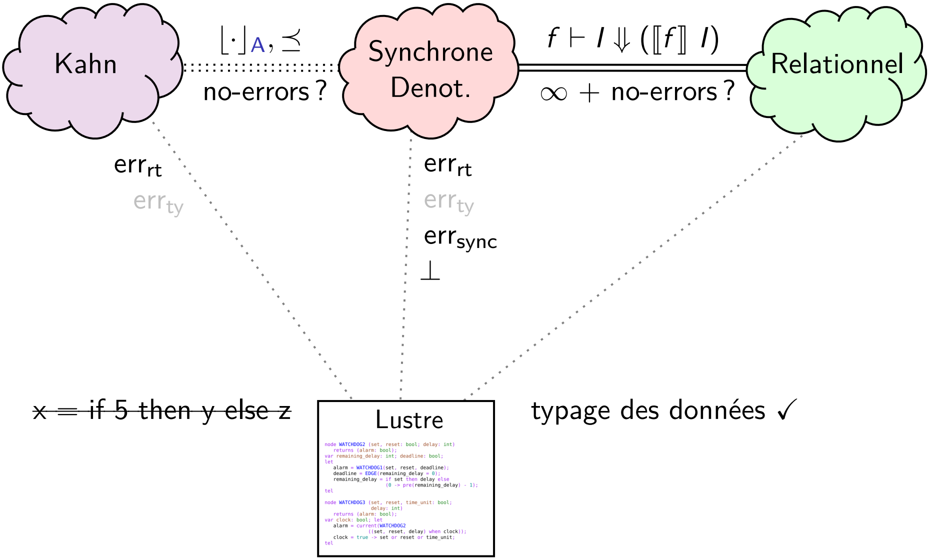
Approche proposée

Traitement des erreurs



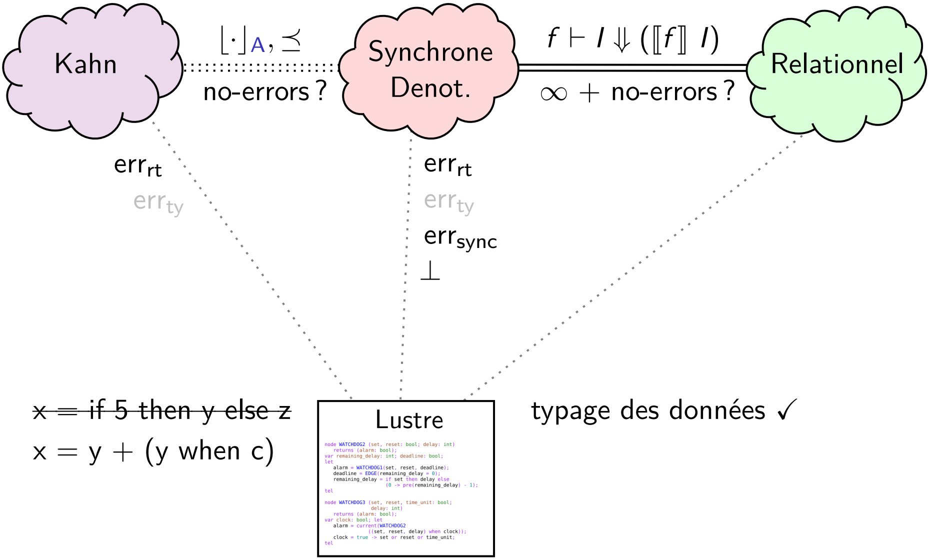
Approche proposée

Traitement des erreurs



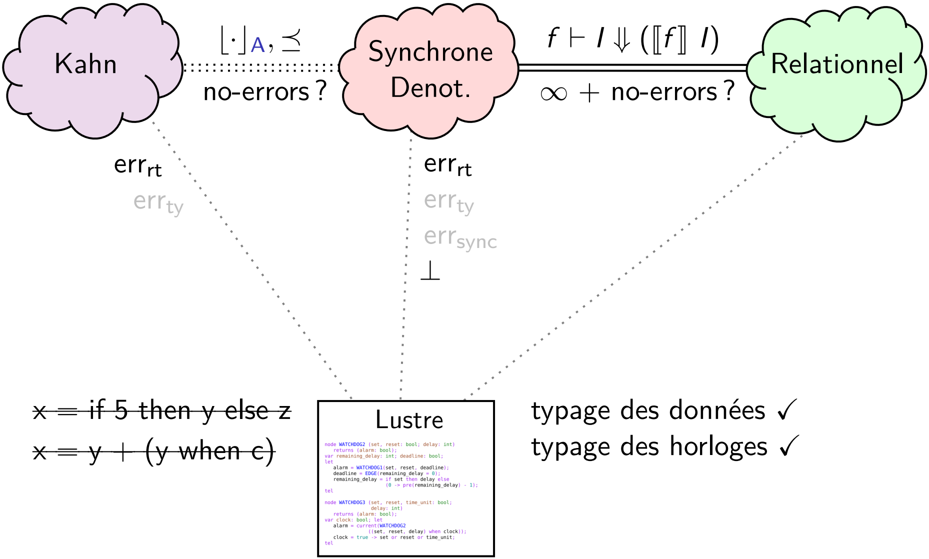
Approche proposée

Traitement des erreurs



Approche proposée

Traitement des erreurs



~~x = if 5 then y else z~~

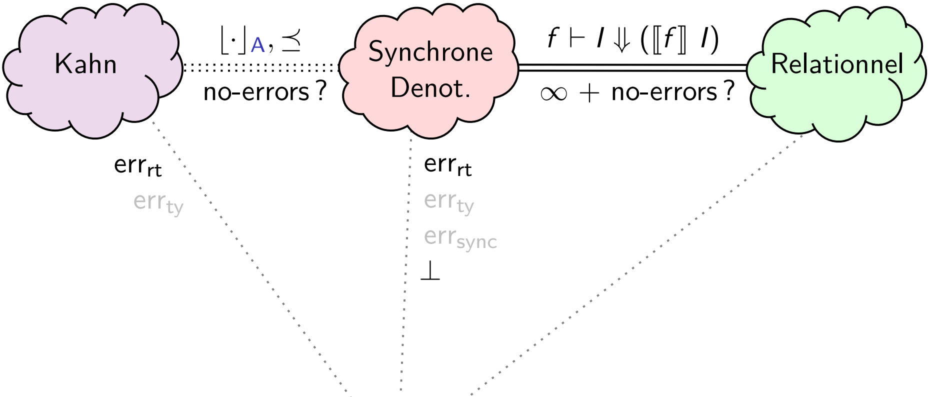
~~x = y + (y when c)~~

typage des données ✓

typage des horloges ✓

Approche proposée

Traitement des erreurs



~~x = if 5 then y else z~~

~~x = y + (y when c)~~

x = x + 5

Lustre

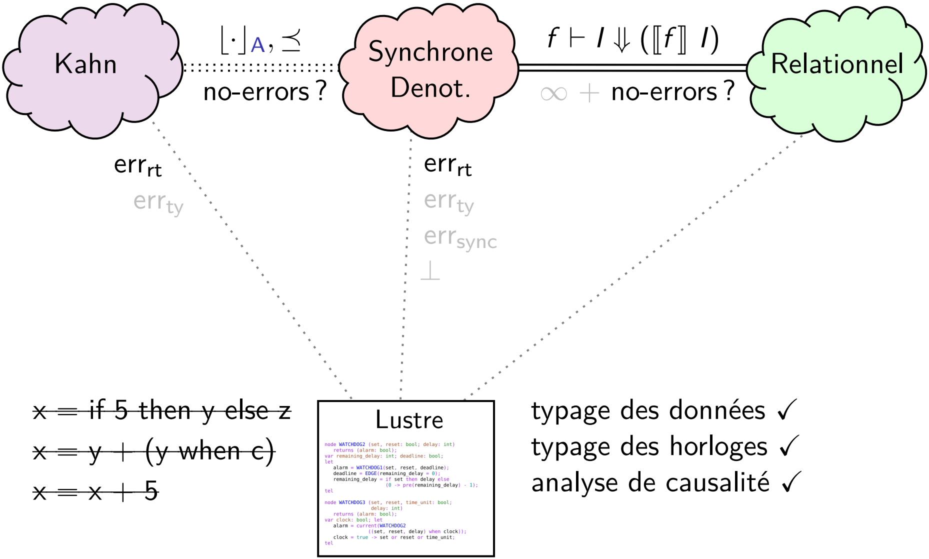
```
node WATCHDOG2 (set, reset: bool; delay: int)
   returns (alarm: bool);
var remaining_delay: int; deadline: bool;
let
   alarm = WATCHDOG1(set, reset, deadline);
   deadline = EDGE(remaining_delay = 0);
   remaining_delay = if set then delay else
                     (0 -> pre(remaining_delay) - 1);
tel

node WATCHDOG3 (set, reset, time_unit: bool;
                delay: int)
   returns (alarm: bool);
var clock: bool; let
   alarm = current(WATCHDOG2
                ((set, reset, delay) when clock));
   clock = true -> set or reset or time_unit;
tel
```

typage des données ✓

typage des horloges ✓

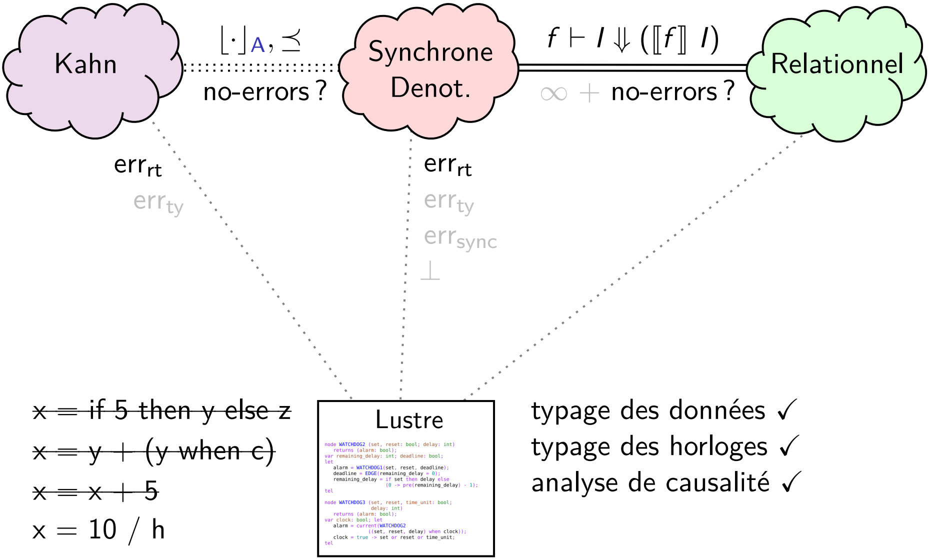
Approche proposée

Traitement des erreurs



Approche proposée

Traitement des erreurs



$\cancel{x = \text{if } 5 \text{ then } y \text{ else } z}$

$\cancel{x = y + (y \text{ when } c)}$

$\cancel{x = x + 5}$

$x = 10 \;/\; h$

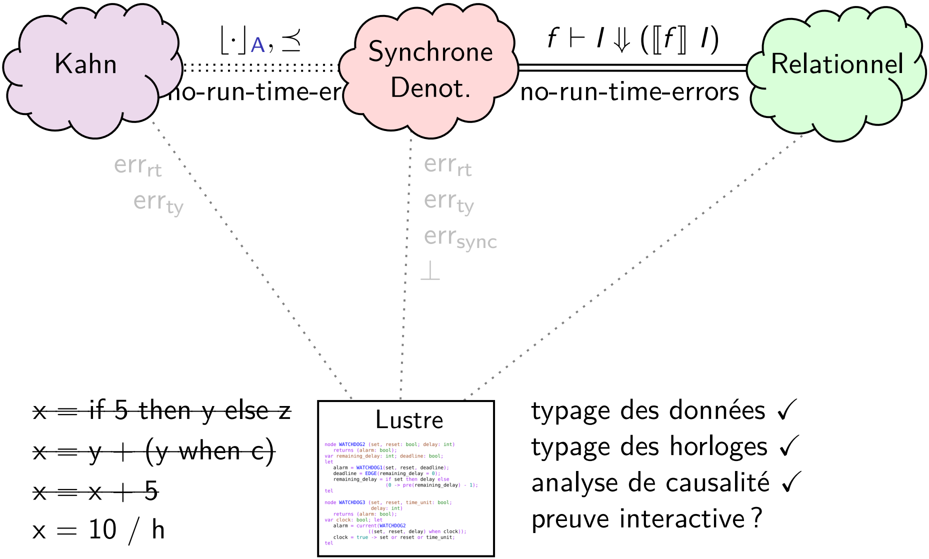
typage des données ✓

typage des horloges ✓

analyse de causalité ✓

Approche proposée

Traitement des erreurs



~~x = if 5 then y else z~~

~~x = y + (y when c)~~

~~x = x + 5~~

x = 10 / h

typage des données ✓

typage des horloges ✓

analyse de causalité ✓

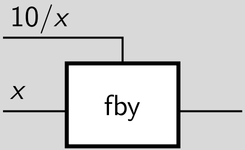
preuve interactive ?

Approche proposée

Traitement des erreurs

Ka

onnel



~~x =~~

~~x =~~

~~x = x + 5~~

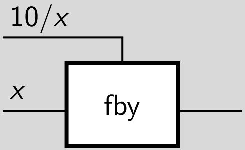
x = 10 / h

analyse de causalité ✓

preuve interactive ?

Approche proposée

Traitement des erreurs



x	5	2	1	0	2	5	$\cdots$
$\text{fby}^{\#}$	2	5	2	1	0	2	$\cdots$

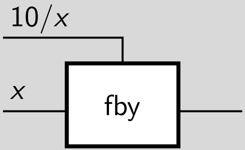
~~x = x + 5~~

x = 10 / h

preuve interactive ?

Approche proposée

Traitement des erreurs



x	5	2	1	0	2	5	$\cdots$
$\mathrm{fby}^{\#}$	2	5	2	1	0	2	$\cdots$
fby	2	5	2	$\mathrm{err}_{\mathrm{rt}}$	$\mathrm{err}_{\mathrm{rt}}$	$\mathrm{err}_{\mathrm{rt}}$	$\cdots$

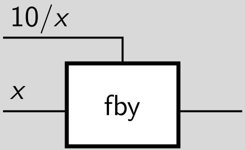
~~x = x + 5~~

x = 10 / h

preuve interactive ?

Approche proposée

Traitement des erreurs



Autres obligations ?

Variations sur $\text{fby}^{\#}$?

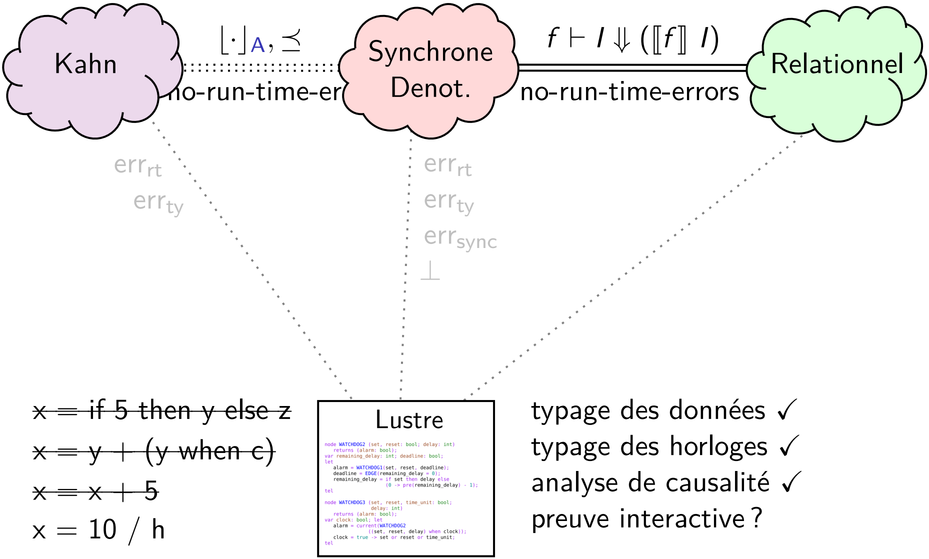
x	5	2	1	0	2	5	$\cdots$
$\text{fby}^{\#}$	2	5	2	1	0	2	$\cdots$
fby	2	5	2	err_{rt}	err_{rt}	err_{rt}	$\cdots$

x = 10 / h

preuve interactive ?

Approche proposée

Traitement des erreurs



~~x = if 5 then y else z~~

~~x = y + (y when c)~~

~~x = x + 5~~

x = 10 / h

typage des données ✓

typage des horloges ✓

analyse de causalité ✓

preuve interactive ?

Analyse statique

Constat : seules quelques opérations bien typées peuvent échouer (cf. CompCert/cfrontend/Cop.v)

Analyse statique

Constat : seules quelques opérations bien typées peuvent échouer (cf. CompCert/cfrontend/Cop.v)

```
n/0              n%0              n >> 64          n << 64
MIN_INT / -1     MIN_INT % -1
(int)NaN         (int)Infty       (int)2^63-1.0    (int)-2^63.0
```

Analyse statique

Constat : seules quelques opérations bien typées peuvent échouer (cf. CompCert/cfrontend/Cop.v)

`n/0`	`n%0`	`n >> 64`	`n << 64`
`MIN_INT / -1`	`MIN_INT % -1`		
`(int)NaN`	`(int)Infty`	`(int)2^63-1.0`	`(int)-2^63.0`

Procédure : interdire ces opérations lorsque l'opérande est inconnue

- check-ops (`4 % x`) = F
- check-ops (`x / 2`) = T
- ...

Analyse statique

Constat : seules quelques opérations bien typées peuvent échouer (cf. CompCert/cfrontend/Cop.v)

`n/0`	`n%0`	`n >> 64`	`n << 64`
`MIN_INT / -1`	`MIN_INT % -1`		
`(int)NaN`	`(int)Infty`	`(int)2^63-1.0`	`(int)-2^63.0`

Procédure : interdire ces opérations lorsque l'opérande est inconnue

- check-ops (`4 % x`) = F
- check-ops (`x / 2`) = T
- …

Théorème : $\forall xs\ f$, si check-ops $(f) =$ T alors no-run-time-errors $f\ xs$

Conclusion

Une correction de la compilation renforcée

Théorème (avant[1])

$$
\begin{array}{rl}
\textbf{si} & \text{compile } f = \text{OK } \mathit{asm} \\
\textbf{et} & \text{welltyped-inputs } f \ \mathit{xs} \\
\textbf{et} & f \vdash \mathit{xs} \Downarrow \mathit{ys} \\
\textbf{alors} & \mathit{asm} \Downarrow \langle \text{Load } (\mathit{xs}(i)) \cdot \text{Store } (\mathit{ys}(i)) \rangle_{i=0}^{\infty}
\end{array}
$$

1. avec machines à états

Conclusion

Une correction de la compilation renforcée

Théorème (avant [1])

$$\begin{array}{rl}
\textbf{si} & \text{compile } f = \text{OK } \mathit{asm} \\
\textbf{et} & \text{welltyped-inputs } f \ \mathit{xs} \\
\textbf{et} & f \vdash \mathit{xs} \Downarrow \mathit{ys} \\
\textbf{alors} & \mathit{asm} \Downarrow \langle \text{Load } (\mathit{xs}(i)) \cdot \text{Store } (\mathit{ys}(i)) \rangle_{i=0}^{\infty}
\end{array}$$

Théorème (après [2])

$$\begin{array}{rl}
\textbf{si} & \text{compile } f = \text{OK } \mathit{asm} \\
\textbf{et} & \text{welltyped-inputs } f \ \mathit{xs} \\
\textbf{et} & \text{no-run-time-errors } f \ \mathit{xs} \\
\textbf{alors} & \exists \mathit{ys}, \ f \vdash \mathit{xs} \Downarrow \mathit{ys} \ \wedge \ \mathit{asm} \Downarrow \langle \text{Load}(\mathit{xs}(i)) \cdot \text{Store}(\mathit{ys}(i)) \rangle_{i=0}^{\infty}
\end{array}$$

1. avec machines à états
2. sans machines à états

Conclusion

Une correction de la compilation renforcée

Théorème (avant[1])

$$\begin{array}{rl} \textbf{si} & \text{compile } f = \text{OK } \mathit{asm} \\ \textbf{et} & \text{welltyped-inputs } f \ \mathit{xs} \\ \textbf{et} & f \vdash \mathit{xs} \Downarrow \mathit{ys} \\ \textbf{alors} & \mathit{asm} \Downarrow \langle \text{Load } (\mathit{xs}(i)) \cdot \text{Store } (\mathit{ys}(i)) \rangle_{i=0}^{\infty} \end{array}$$

Théorème (après[2])

$$\begin{array}{rl} \textbf{si} & \text{compile } f = \text{OK } \mathit{asm} \\ \textbf{et} & \text{welltyped-inputs } f \ \mathit{xs} \\ \textbf{et} & \text{check-ops } (f) = \text{T} \\ \textbf{alors} & \exists \mathit{ys},\ f \vdash \mathit{xs} \Downarrow \mathit{ys} \ \wedge \ \mathit{asm} \Downarrow \langle \text{Load}(\mathit{xs}(i)) \cdot \text{Store}(\mathit{ys}(i)) \rangle_{i=0}^{\infty} \end{array}$$

---

1. avec machines à états
2. sans machines à états

Conclusion

Nous avons défini :

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus
- une modélisation précise des erreurs dans le contexte d'une évaluation par point fixe, dans Coq

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus
- une modélisation précise des erreurs dans le contexte d'une évaluation par point fixe, dans Coq
- des critères de sa correspondance avec le modèle de Kahn

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus
- une modélisation précise des erreurs dans le contexte d'une évaluation par point fixe, dans Coq
- des critères de sa correspondance avec le modèle de Kahn
- une analyse statique pour valider certaines exécutions

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus
- une modélisation précise des erreurs dans le contexte d'une évaluation par point fixe, dans Coq
- des critères de sa correspondance avec le modèle de Kahn
- une analyse statique pour valider certaines exécutions

Il reste à faire :

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus
- une modélisation précise des erreurs dans le contexte d'une évaluation par point fixe, dans Coq
- des critères de sa correspondance avec le modèle de Kahn
- une analyse statique pour valider certaines exécutions

Il reste à faire :

- une extension aux automates hiérarchiques

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus
- une modélisation précise des erreurs dans le contexte d'une évaluation par point fixe, dans Coq
- des critères de sa correspondance avec le modèle de Kahn
- une analyse statique pour valider certaines exécutions

Il reste à faire :

- une extension aux automates hiérarchiques
- une analyse statique plus élaborée

Conclusion

Nous avons défini :

- une sémantique dénotationnelle synchrone pour un langage à flots de données avec réinitialisation modulaire, dans le cadre du compilateur Vélus
- une modélisation précise des erreurs dans le contexte d'une évaluation par point fixe, dans Coq
- des critères de sa correspondance avec le modèle de Kahn
- une analyse statique pour valider certaines exécutions

Il reste à faire :

- une extension aux automates hiérarchiques
- une analyse statique plus élaborée
- affiner les critères pour pouvoir raisonner sur le modèle de Kahn