

Une sémantique dénotationnelle pour un compilateur synchrone vérifié

Paul Jeanmaire

Thèse encadrée par Timothy Bourke et Marc Pouzet

à l'ENS, Inria, équipe Parkas

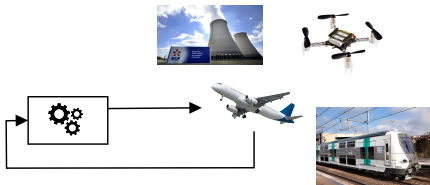
20 décembre 2024



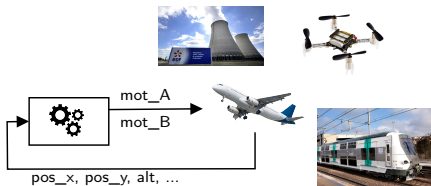
Programmation des systèmes embarqués



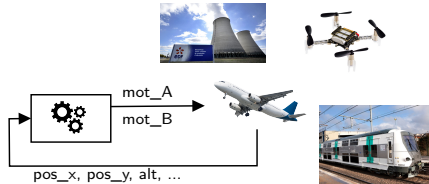
Programmation des systèmes embarqués



Programmation des systèmes embarqués

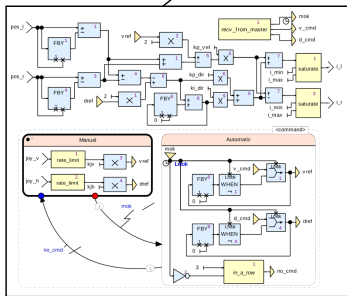
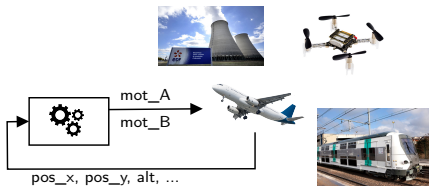


Programmation des systèmes embarqués

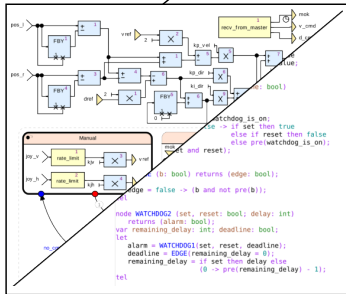
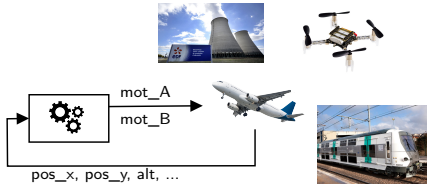


```
# L'adresse label est l'adresse de la fonction
label:
    .align 16
    .global FullStepsCount_down
FullStepsCount_down:
    .cfi_startproc
    subq $8, %rip
    .cfi_adjust_cfa_offset 8
    leaq 16(%rip), %rax
    movq %rax, %rsi
    andl $15, %rsi
    movl 8(%rsi), %eax
    testl %eax, %eax
    jnz 15(%rip), %eax
    movl 4(%rsi), %eax
    .L1000:
    sarl %eax, %eax
    movl %eax, 8(%rsi)
    leaq 16(%rip), %rax
    movl %rax, 4(%rsi)
    movq %rax, %rsi
    addq $8, %rip
    ret
    .cfi_endproc
.type FullStepsCount_down, @function
.size FullStepsCount_down, . - FullStepsCount_down
    .align 16
    .global FullStepsCount_down
FullStepsCount_down:
    .cfi_startproc
    subq $8, %rip
    .cfi_adjust_cfa_offset 8
    leaq 16(%rip), %rax
    movq %rax, %rsi
    andl $15, %rsi
    sarl %eax, %eax
    movl %eax, 8(%rsi)
    .L1000:
    sarl %eax, %eax
    movl %eax, 4(%rsi)
    .cfi_endproc
.type FullStepsCount_down, @function
.size FullStepsCount_down, . - FullStepsCount_down
```

Programmation des systèmes embarqués

[illegible]

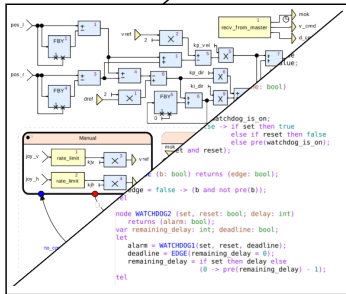
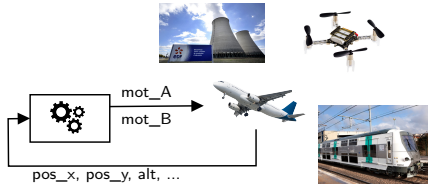
Programmation des systèmes embarqués



```
# Summary: ./CMakeLists.txt

# 1. add_executable
add_executable(
    # 1.1. target
    # 1.2. source files
    FastestSegmentDown
    # 1.3. headers
    # 1.4. libraries
    # 1.5. link options
    # 1.6. libraries
    # 1.7. libraries
    # 1.8. libraries
    # 1.9. libraries
    # 1.10. libraries
    # 1.11. libraries
    # 1.12. libraries
    # 1.13. libraries
    # 1.14. libraries
    # 1.15. libraries
    # 1.16. libraries
    # 1.17. libraries
    # 1.18. libraries
    # 1.19. libraries
    # 1.20. libraries
    # 1.21. libraries
    # 1.22. libraries
    # 1.23. libraries
    # 1.24. libraries
    # 1.25. libraries
    # 1.26. libraries
    # 1.27. libraries
    # 1.28. libraries
    # 1.29. libraries
    # 1.30. libraries
    # 1.31. libraries
    # 1.32. libraries
    # 1.33. libraries
    # 1.34. libraries
    # 1.35. libraries
    # 1.36. libraries
    # 1.37. libraries
    # 1.38. libraries
    # 1.39. libraries
    # 1.40. libraries
    # 1.41. libraries
    # 1.42. libraries
    # 1.43. libraries
    # 1.44. libraries
    # 1.45. libraries
    # 1.46. libraries
    # 1.47. libraries
    # 1.48. libraries
    # 1.49. libraries
    # 1.50. libraries
    # 1.51. libraries
    # 1.52. libraries
    # 1.53. libraries
    # 1.54. libraries
    # 1.55. libraries
    # 1.56. libraries
    # 1.57. libraries
    # 1.58. libraries
    # 1.59. libraries
    # 1.60. libraries
    # 1.61. libraries
    # 1.62. libraries
    # 1.63. libraries
    # 1.64. libraries
    # 1.65. libraries
    # 1.66. libraries
    # 1.67. libraries
    # 1.68. libraries
    # 1.69. libraries
    # 1.70. libraries
    # 1.71. libraries
    # 1.72. libraries
    # 1.73. libraries
    # 1.74. libraries
    # 1.75. libraries
    # 1.76. libraries
    # 1.77. libraries
    # 1.78. libraries
    # 1.79. libraries
    # 1.80. libraries
    # 1.81. libraries
    # 1.82. libraries
    # 1.83. libraries
    # 1.84. libraries
    # 1.85. libraries
    # 1.86. libraries
    # 1.87. libraries
    # 1.88. libraries
    # 1.89. libraries
    # 1.90. libraries
    # 1.91. libraries
    # 1.92. libraries
    # 1.93. libraries
    # 1.94. libraries
    # 1.95. libraries
    # 1.96. libraries
    # 1.97. libraries
    # 1.98. libraries
    # 1.99. libraries
    # 1.100. libraries
    # 1.101. libraries
    # 1.102. libraries
    # 1.103. libraries
    # 1.104. libraries
    # 1.105. libraries
    # 1.106. libraries
    # 1.107. libraries
    # 1.108. libraries
    # 1.109. libraries
    # 1.110. libraries
    # 1.111. libraries
    # 1.112. libraries
    # 1.113. libraries
    # 1.114. libraries
    # 1.115. libraries
    # 1.116. libraries
    # 1.117. libraries
    # 1.118. libraries
    # 1.119. libraries
    # 1.120. libraries
    # 1.121. libraries
    # 1.122. libraries
    # 1.123. libraries
    # 1.124. libraries
    # 1.125. libraries
    # 1.126. libraries
    # 1.127. libraries
    # 1.128. libraries
    # 1.129. libraries
    # 1.130. libraries
    # 1.131. libraries
    # 1.132. libraries
    # 1.133. libraries
    # 1.134. libraries
    # 1.135. libraries
    # 1.136. libraries
    # 1.137. libraries
    # 1.138. libraries
    # 1.139. libraries
    # 1.140. libraries
    # 1.141. libraries
    # 1.142. libraries
    # 1.143. libraries
    # 1.144. libraries
    # 1.145. libraries
    # 1.146. libraries
    # 1.147. libraries
    # 1.148. libraries
    # 1.149. libraries
    # 1.150. libraries
    # 1.151. libraries
    # 1.152. libraries
    # 1.153. libraries
    # 1.154. libraries
    # 1.155. libraries
    # 1.156. libraries
    # 1.157. libraries
    # 1.158. libraries
    # 1.159. libraries
    # 1.160. libraries
    # 1.161. libraries
    # 1.162. libraries
    # 1.163. libraries
    # 1.164. libraries
    # 1.165. libraries
    # 1.166. libraries
    # 1.167. libraries
    # 1.168. libraries
    # 1.169. libraries
    # 1.170. libraries
    # 1.171. libraries
    # 1.172. libraries
    # 1.173. libraries
    # 1.174. libraries
    # 1.175. libraries
    # 1.176. libraries
    # 1.177. libraries
    # 1.178. libraries
    # 1.179. libraries
    # 1.180. libraries
    # 1.181. libraries
    # 1.182. libraries
    # 1.183. libraries
    # 1.184. libraries
    # 1.185. libraries
    # 1.186. libraries
    # 1.187. libraries
    # 1.188. libraries
    # 1.189. libraries
    # 1.190. libraries
    # 1.191. libraries
    # 1.192. libraries
    # 1.193. libraries
    # 1.194. libraries
    # 1.195. libraries
    # 1.196. libraries
    # 1.197. libraries
    # 1.198. libraries
    # 1.199. libraries
    # 1.200. libraries
    # 1.201. libraries
    # 1.202. libraries
    # 1.203. libraries
    # 1.204. libraries
    # 1.205. libraries
    # 1.206. libraries
    # 1.207. libraries
    # 1.208. libraries
    # 1.209. libraries
    # 1.210. libraries
    # 1.211. libraries
    # 1.212. libraries
    # 1.213. libraries
    # 1.214. libraries
    # 1.215. libraries
    # 1.216. libraries
    # 1.217. libraries
    # 1.218. libraries
    # 1.219. libraries
    # 1.220. libraries
    # 1.221. libraries
    # 1.222. libraries
    # 1.223. libraries
    # 1.224. libraries
    # 1.225. libraries
    # 1.226. libraries
    # 1.227. libraries
    # 1.228. libraries
    # 1.229. libraries
    # 1.230. libraries
    # 1.231. libraries
    # 1.232. libraries
    # 1.233. libraries
    # 1.234. libraries
    # 1.235. libraries
    # 1.236. libraries
    # 1.237. libraries
    # 1.238. libraries
    # 1.239. libraries
    # 1.240. libraries
    # 1.241. libraries
    # 1.242. libraries
    # 1.243. libraries
    # 1.244. libraries
    # 1.2
```

Programmation des systèmes embarqués



compilation

[illegible]

Compilation synchrone vérifiée

Lustre

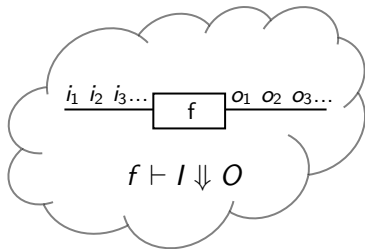
```
node WATCHDOG (set, reset: bool; delay: int)
  returns (alarm: bool);
  var remaining_delay: int; deadline: bool;
  set
  {
    alarm = WATCHDOG(set, reset, deadline);
    deadline = (delay < remaining_delay & 0);
    remaining_delay = 0 set then delay else
      (0 -> pre(remaining_delay) - 1);
  }
  set
  {
    node WATCHDOG (set, reset, time_unit: bool);
    delay := set;
  }
  var clock: bool; let
  alarm = current(WATCHDOG2
    (set, reset, delay) when clock);
  clock = true -> set or reset or time_unit;
  set
```

Vélus + CompCert

Assembleur

```
functio watchdog_alarm:
  var remaining_delay: int;
  var deadline: bool;
  set
  {
    alarm = WATCHDOG(set, reset, deadline);
    deadline = (delay < remaining_delay & 0);
    remaining_delay = 0 set then delay else
      (0 -> pre(remaining_delay) - 1);
  }
  set
  {
    node WATCHDOG (set, reset, time_unit: bool);
    delay := set;
  }
  var clock: bool; let
  alarm = current(WATCHDOG2
    (set, reset, delay) when clock);
  clock = true -> set or reset or time_unit;
  set
```

Compilation synchrone vérifiée



Lustre

```
node MATCH062 {set, reset: bool; delay: int;
  returns {alarm: bool};
  var remaining: delay; {set; deadline: bool};
  set
    alarm = MATCH0061 {set, reset, deadline:
      deadline = EDGE {remaining delay = 0};
      remaining delay = 0 if set then delay else
        (0 -> pre{remaining delay} - 1);
  }
}

node MATCH063 {set, reset, time_unit: bool;
  returns {alarm: bool};
  var clock: bool; {set;
  alarm = current {MATCH062
    {set, reset, delay} when clock;
  }
  }
  {clock = true -> set or reset or time_unit;
}
```

Vélus + CompCert

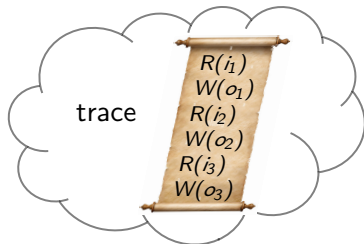
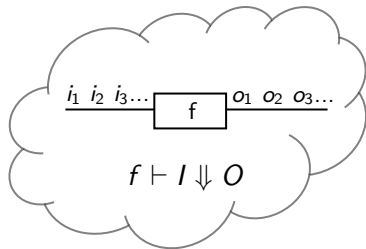
Assembleur

```

funtest@pcount_down:
.cfi startproc
subq    $8, %rsp
.cfi adjust_cfi_offset
leaq    16(%rsp), %rax
movq    %rax, 0(%rsp)
testl   %eax, %eax
jne     .L100
movzbl  0(%rdi), %eax
testl   %eax, %eax
jne     .L100
movl    4(%rdi), %eax
.L100:
xorl    %eax, %eax
movb    %al, 0(%rdi)
leal    -1(%eax), %eax
movl    %eax, 4(%rdi)
movq    %rax, %rax
addq    $8, %rsp

```

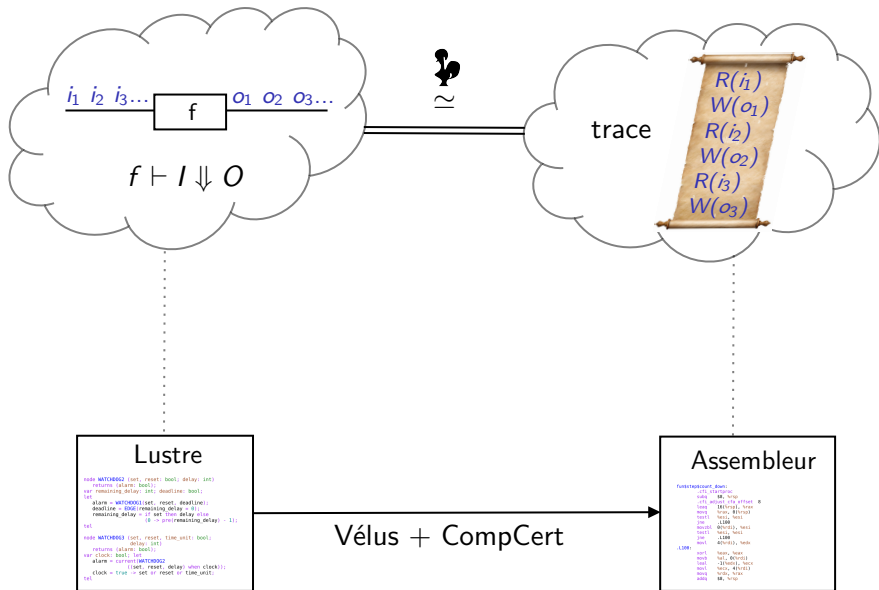
Compilation synchrone vérifiée



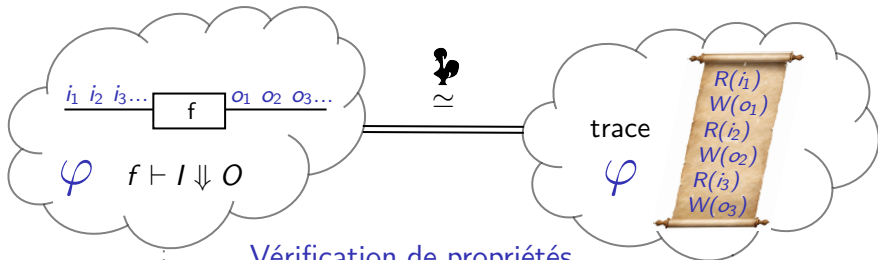
Vélus + CompCert



Compilation synchrone vérifiée



Compilation synchrone vérifiée



Vérification de propriétés

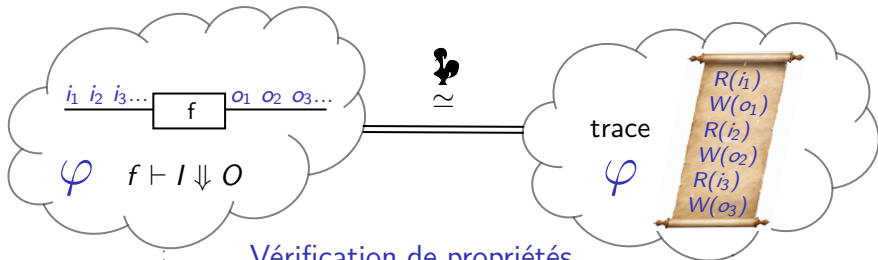
- ▶ interactive/automatique



Vélus + CompCert



Compilation synchrone vérifiée



Vérification de propriétés

- ▶ interactive/automatique
- ▶ lisibilité?



Vélus + CompCert



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.

```

```

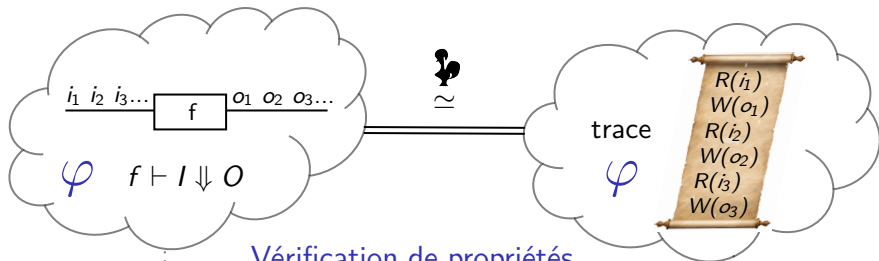
end; ys.

node
  (set
    remaining_delay: int; deadline: bool;
  set
    alarm = WATCHDOG1 (set, reset, deadline:
      deadline := (0 < remaining_delay & 0);
      remaining_delay := if set then delay else
        ($ -> pre(remaining_delay) - 1);
    set
  node WATCHDOG2 (set, reset, time_unit: bool;
    delay: int)
    return (alarm, bool);
  set clock: bool; int
  alarm = current(WATCHDOG2
    (set, reset, delay) when clock);
  clock = true -> set or reset or time_unit.
end

```

Vélus + CompCert

Compilation synchrone vérifiée



Vérification de propriétés

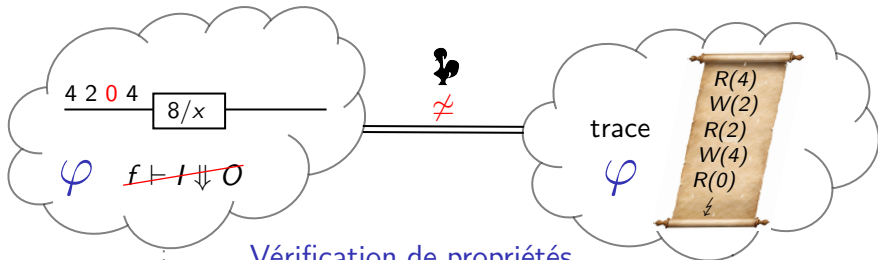
- ▶ interactive/automatique
- ▶ lisibilité?



Vélus + CompCert



Compilation synchrone vérifiée



Vérification de propriétés

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

Lustre

```
node MATCH0662 {set, reset, bool; delay: int;
  returns (alarm: bool);
  var remaining_delay: int;
  set deadline: bool;
  alarm = WATCHDOG1 {set, reset, deadline:
    deadline = EDGE {remaining_delay = 0};
    remaining_delay = 0 if set then delay else
      remaining_delay - 1;
  }
}

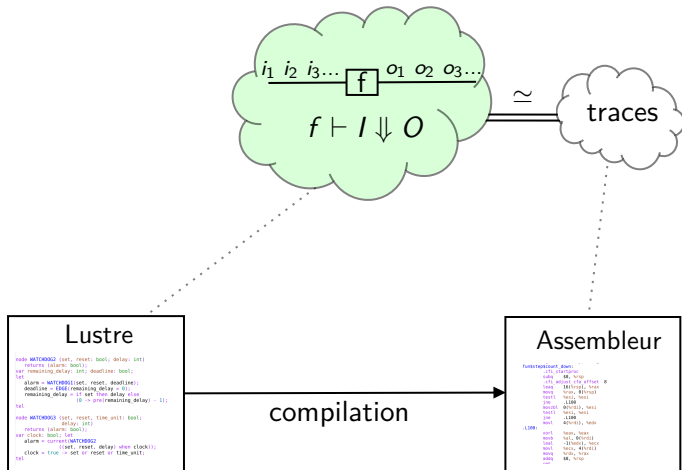
node MATCH0663 {set, reset, time_unit: bool;
  returns (alarm: bool);
  var clock: bool;
  alarm = current {MATCH0662
    {set, reset, bool; delay: int;
      returns (alarm: bool);
      var remaining_delay: int;
      set deadline: bool;
      alarm = WATCHDOG1 {set, reset, deadline:
        deadline = EDGE {remaining_delay = 0};
        remaining_delay = 0 if set then delay else
          remaining_delay - 1;
      }
    }
  }
  (clock = true → set or reset or time_unit);
}
```

Vélus + CompCert

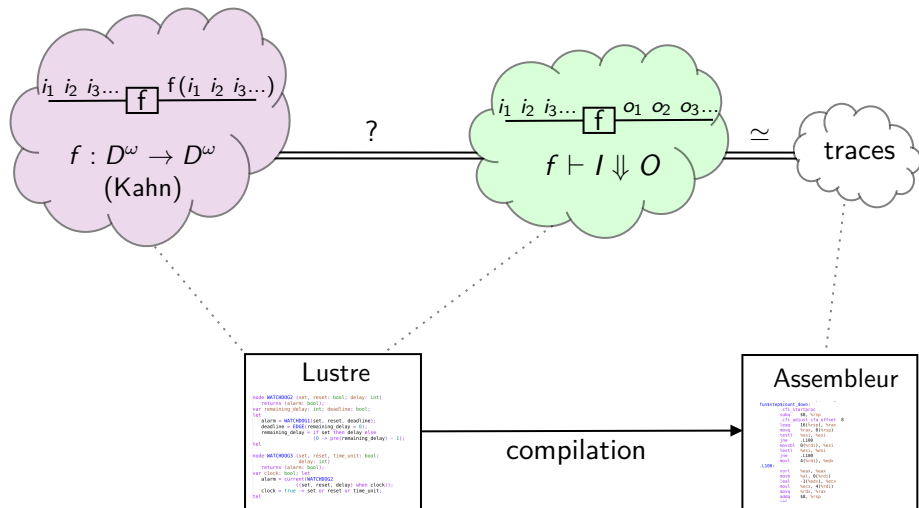
Assembleur

```
funcstepscount_down:
    cfi_startproc
    addq    $8, %rsp
    cfi_adjust_cfp_offset
    leaq    18(%rsp), %rax
    movq    %rax, 0(%rsp)
    testl   %eax, %eax
    jne     .L100
    movzbl  0(%rdi), %eax
    testl   %eax, %eax
    jne     .L100
    movl    4(%rdi), %eax
.L100:
    ret     %eax, %eax
    movb    %al, 0(%rdi)
    leal    -2(%eax), %eax
    movl    %eax, 4(%rdi)
    movq    %rax, %rax
    addq    $8, %rsp
```

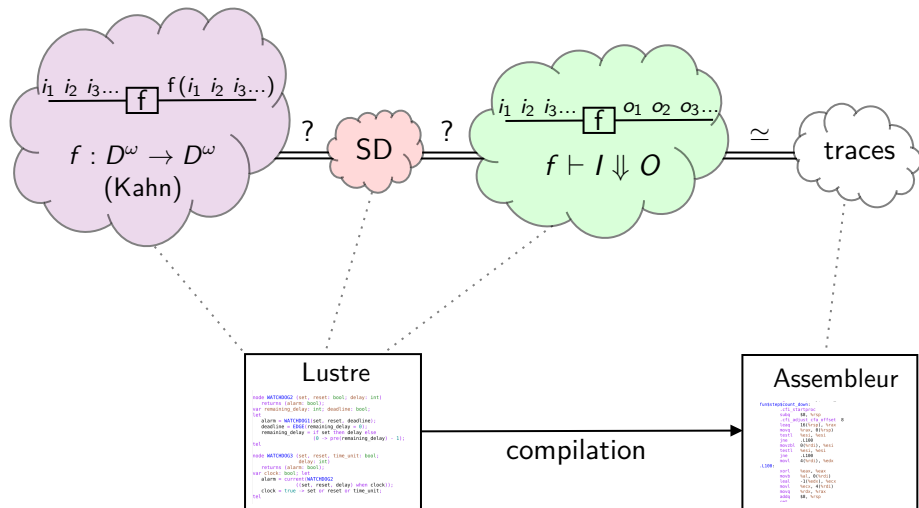
Approche proposée



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^∞ $D := A \mid v$
mécanique			
erreurs			

Trois sémantiques à flots de données

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

Trois sémantiques à flots de données

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

Trois sémantiques à flots de données

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

Trois sémantiques à flots de données

	Kahn	Synchrone dénot.	Relationnelle
flots	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := A \mid v$	D^∞ $D := A \mid v$
mécanique	dénotationnelle (ordre préfixe), $\perp = \epsilon$ 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	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := A \mid v$	D^∞ $D := A \mid v$
mécanique	dénotationnelle (ordre préfixe), $\perp = \epsilon$ 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	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := A \mid v$	D^∞ $D := A \mid v$
mécanique	dénotationnelle (ordre préfixe), $\perp = \epsilon$ 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	$D^\omega = D^* \cup D^\infty$ $D := v$	$D^\omega = D^* \cup D^\infty$ $D := A \mid v$	D^∞ $D := A \mid v$
mécanique	dénotationnelle (ordre préfixe), $\perp = \epsilon$ 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 : $(D \rightarrow \mathbb{B}) \rightarrow D^\infty \rightarrow D^\omega$

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, $(\omega\text{-})$ CPO, continuité, points fixes ...

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

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

$$\forall F \ P, \text{ admissible } P \rightarrow P \perp \rightarrow (\forall x, P \ x \rightarrow P(F \ x)) \rightarrow P(\text{fixp } F)$$

Bibliothèque de flots

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

Domaines de Scott, $(\omega\text{-})\text{CPO}$, continuité, points fixes ...

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

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

$$\forall F \ P, \text{ admissible } P \rightarrow P \perp \rightarrow (\forall x, P \ x \rightarrow P(F \ x)) \rightarrow P(\text{fixp } F)$$

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, $(\omega\text{-})\text{CPO}$, continuité, points fixes ...

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

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

$$\forall F \ P, \text{ admissible } P \rightarrow P \perp \rightarrow (\forall x, P \ x \rightarrow P(F \ x)) \rightarrow P(\text{fixp } F)$$

Construction de D^ω

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

```
CoFixpoint bot := Tau bot.
```

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

Bibliothèque de flots

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

Domaines de Scott, $(\omega\text{-})\text{CPO}$, continuité, points fixes ...

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

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

$$\forall F \ P, \text{ admissible } P \rightarrow P \perp \rightarrow (\forall x, P \ x \rightarrow P(F \ x)) \rightarrow P(\text{fixp } F)$$

Construction de D^ω

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

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

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

```
CoFixpoint bot := Tau bot.
```

Bibliothèque de flots

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

Domaines de Scott, $(\omega\text{-})\text{CPO}$, continuité, points fixes ...

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

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

$$\forall F \ P, \text{ admissible } P \rightarrow P \perp \rightarrow (\forall x, P \ x \rightarrow P(F \ x)) \rightarrow P(\text{fixp } F)$$

Construction de D^ω

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

```
CoFixpoint bot := Tau bot.
```

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

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

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

Bibliothèque de flots

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

Domaines de Scott, $(\omega\text{-})\text{CPO}$, continuité, points fixes ...

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

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

$$\forall F \ P, \text{ admissible } P \rightarrow P \perp \rightarrow (\forall x, P \ x \rightarrow P(F \ x)) \rightarrow P(\text{fixp } F)$$

Construction de D^ω

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

```
CoFixpoint bot := Tau bot.
```

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

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

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

$$\preceq a \cdot b \cdot c \cdot \tau \cdot \tau \cdot \tau \cdot \tau \cdots$$

Bibliothèque de flots (suite)

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

$\text{filter} : (A \rightarrow \mathbb{B}) \rightarrow A^\omega \rightarrow_c A^\omega$

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

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

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

Bibliothèque de flots (suite)

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

$\text{filter} : (A \rightarrow \mathbb{B}) \rightarrow A^\omega \rightarrow_c A^\omega$

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

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

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

$\text{map} : (A \rightarrow B) \rightarrow A^\omega \rightarrow_c B^\omega$

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

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

Bibliothèque de flots (suite)

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

$\text{filter} : (A \rightarrow \mathbb{B}) \rightarrow A^\omega \rightarrow_c A^\omega$

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

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

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

$\text{map} : (A \rightarrow B) \rightarrow A^\omega \rightarrow_c B^\omega$

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

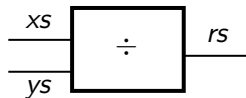
$\text{map } f \perp \simeq \perp$

$\text{zip} : (A \rightarrow B \rightarrow C) \rightarrow A^\omega \rightarrow_c B^\omega \rightarrow_c C^\omega$

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

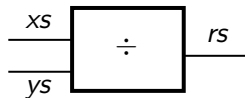
$\text{zip } f s \perp \simeq \text{zip } f \perp s \simeq \perp$

Construction : opération combinatoire



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

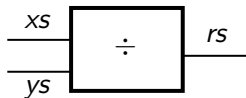
Construction : opération combinatoire



<i>xs</i>	<i>A</i>	10	10	<i>A</i>	10	10	...
<i>ys</i>	<i>A</i>	10	5	<i>A</i>	2	1	...
<i>rs</i>	<i>A</i>	1	2	<i>A</i>	5	10	...

$$\begin{array}{c}
 \text{LIFT } xs \ ys \ rs \\
 \hline \hline
 \text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs) \\
 \\
 \text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r \\
 \hline \hline
 \text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)
 \end{array}$$

Construction : opération combinatoire

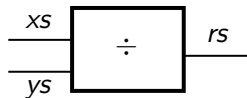


<i>xs</i>	<i>A</i>	10	10	<i>A</i>	10	10	...
<i>ys</i>	<i>A</i>	10	5	<i>A</i>	2	1	...
<i>rs</i>	<i>A</i>	1	2	<i>A</i>	5	10	...

lift : $\text{Str} \rightarrow_c \text{Str} \rightarrow_c \text{Str} :=$
zip

$$\begin{array}{c}
 \text{LIFT } xs \ ys \ rs \\
 \hline \hline
 \text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs) \\
 \\
 \text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r \\
 \hline \hline
 \text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)
 \end{array}$$

Construction : opération combinatoire



<i>xs</i>	<i>A</i>	10	10	<i>A</i>	10	10	...
<i>ys</i>	<i>A</i>	10	5	<i>A</i>	2	1	...
<i>rs</i>	<i>A</i>	1	2	<i>A</i>	5	10	...

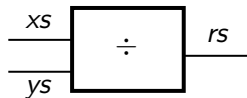
lift : Str \rightarrow_c Str \rightarrow_c Str :=

zip (λ *A*, *A* \rightarrow *A*

)

$$\begin{array}{c}
 \text{LIFT } xs \ ys \ rs \\
 \hline \hline
 \text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs) \\
 \\
 \text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r \\
 \hline \hline
 \text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)
 \end{array}$$

Construction : opération combinatoire



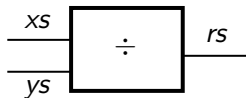
<i>xs</i>	<i>A</i>	10	10	<i>A</i>	10	10	...
<i>ys</i>	<i>A</i>	10	5	<i>A</i>	2	1	...
<i>rs</i>	<i>A</i>	1	2	<i>A</i>	5	10	...

```

lift  : Str →c Str →c Str :=
zip (λ A, A → A
    | v1, v2 → (match v1 ÷ v2 with
                  | Some r → r
                  | None  → errrt)
    )
  
```

$$\begin{array}{c}
 \text{LIFT } xs \ ys \ rs \\
 \hline \hline
 \text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs) \\
 \\
 \text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r \\
 \hline \hline
 \text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)
 \end{array}$$

Construction : opération combinatoire



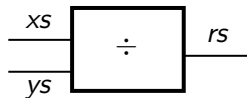
<i>xs</i>	<i>A</i>	10	10	<i>A</i>	10	10	...
<i>ys</i>	<i>A</i>	10	5	<i>A</i>	2	1	...
<i>rs</i>	<i>A</i>	1	2	<i>A</i>	5	10	...

```

lift  : Str →c Str →c Str :=
zip (λ A, A → A
    | v1, v2 → (match v1 ÷ v2 with
                    | Some r → r
                    | None → errrt)
    | v, A | A, v → errsync
    )
    
```

$$\begin{array}{c}
 \text{LIFT } xs \ ys \ rs \\
 \hline \hline
 \text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs) \\
 \\
 \text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r \\
 \hline \hline
 \text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)
 \end{array}$$

Construction : opération combinatoire



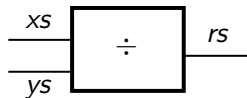
<i>xs</i>	<i>A</i>	10	10	<i>A</i>	10	10	...
<i>ys</i>	<i>A</i>	10	5	<i>A</i>	2	1	...
<i>rs</i>	<i>A</i>	1	2	<i>A</i>	5	10	...

```

lift  : Str →c Str →c Str :=
zip (λ A, A → A
    | v1, v2 → (match v1 ÷ v2 with
                    | Some r → r
                    | None → errrt)
    | v, A | A, v → errsync
    | err, _ | _, err → err)
  
```

$$\begin{array}{c}
 \text{LIFT } xs \ ys \ rs \\
 \hline \hline
 \text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs) \\
 \\
 \text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r \\
 \hline \hline
 \text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)
 \end{array}$$

Construction : opération combinatoire



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

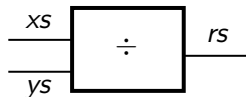
```

lift  : Str →c Str →c Str :=
zip (λ A, A → A
    | v1, v2 → (match v1 ÷ v2 with
                  | Some r → r
                  | None → errrt)
    | v, A | A, v → errsync
    | err, _ | _, err → err)
  
```

$$\begin{array}{c}
 \text{LIFT } xs \ ys \ rs \\
 \hline \hline
 \text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs) \\
 \\
 \text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r \\
 \hline \hline
 \text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)
 \end{array}$$

🐼 **Théorème** : si lift xs ys est sans erreurs, alors LIFT xs ys (lift xs ys)

Construction : opération combinatoire





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

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

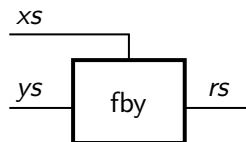
$$\frac{\text{LIFT } xs \ ys \ rs}{\text{LIFT } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs)}$$

$$\frac{\text{LIFT } xs \ ys \ rs \quad v_1 \div v_2 = \text{Some } r}{\text{LIFT } (v_1 \cdot xs) \ (v_2 \cdot ys) \ (r \cdot rs)}$$

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

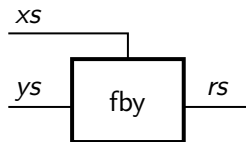
 **Théorème** : et aussi, $[\text{lift } xs \ ys]_A \simeq \text{lift}^\# [xs]_A [ys]_A$

Construction : délai initialisé



<i>xs</i>	5	5	5	5	5	5	...
<i>ys</i>	6	7	8	9	10	11	...
<i>rs</i>	5	6	7	8	9	10	...

Construction : délai initialisé



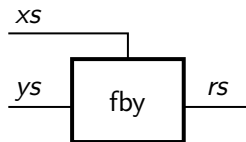
xs	5	5	5	5	5	5	...
ys	6	7	8	9	10	11	...
rs	5	6	7	8	9	10	...

$$\text{fby}^\# : \text{Str} \rightarrow_c \text{Str} \rightarrow_c \text{Str}$$

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

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

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	...

$$\text{fby}^\# : \text{Str} \rightarrow_c \text{Str} \rightarrow_c \text{Str}$$

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

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

$$\text{FBY } xs \ ys \ rs$$

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

$$\text{FBY}_1 \ v_2 \ xs \ ys \ rs$$

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

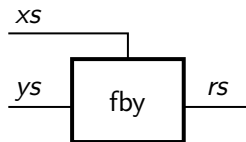
$$\text{FBY}_1 \ v \ xs \ ys \ rs$$

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

$$\text{FBY}_1 \ v_2 \ xs \ ys \ rs$$

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

Construction : délai initialisé



<i>xs</i>	<i>A</i>	5	<i>A</i>	5	5	5	5	...
<i>ys</i>	<i>A</i>	6	<i>A</i>	7	8	9	10	...
<i>rs</i>	<i>A</i>	5	<i>A</i>	6	7	8	9	...

$$\text{fby}^\# : \text{Str} \rightarrow_c \text{Str} \rightarrow_c \text{Str}$$

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

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

Synchrone dénotationnel

- quelles erreurs ?
- $x = 0 \text{ fby } (x + 1)$?

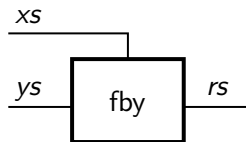
$$\frac{\text{FBY } xs \ ys \ rs}{\text{FBY } (A \cdot xs) \ (A \cdot ys) \ (A \cdot rs)}$$

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

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

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

Construction : délai initialisé



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

$$\text{fby} (A \cdot xs) ys := A \cdot \text{fby}_A xs ys$$

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

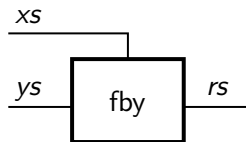
$$\text{fby}_A xs (A \cdot ys) := \text{fby} xs ys$$

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

$$\text{fby}_1 v (A \cdot xs) ys := A \cdot \text{fby}'_1 (\text{Some } v) xs ys$$

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

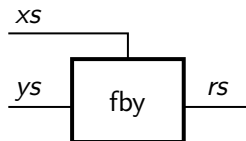
Construction : délai initialisé



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

$$\text{fby } (A \cdot xs) \text{ } ys := A \cdot \text{fby}_A \text{ } xs \text{ } ys$$
$$\text{fby } (v \cdot xs) \text{ } ys := v \cdot \text{fby}'_1 \text{ None } xs \text{ } ys$$
$$\text{fby } (\text{err} \cdot xs) \text{ } ys := \text{err} \cdot \text{map } (\lambda x. \text{err}) \text{ } xs$$
$$\text{fby}_A \text{ } xs \text{ } (A \cdot ys) := \text{fby } xs \text{ } ys$$
$$\text{fby}_A \text{ } xs \text{ } (\text{err} \cdot ys) := \text{map } (\lambda x. \text{err}) \text{ } xs$$
$$\text{fby}_A \text{ } xs \text{ } (v \cdot ys) := \text{map } (\lambda x. \text{err}_{\text{sync}}) \text{ } xs$$
$$\text{fby}'_1 \text{ None } xs \text{ } (v \cdot ys) := \text{fby}_1 \text{ } v \text{ } xs \text{ } ys$$
$$\text{fby}'_1 (\text{Some } v) \text{ } xs \text{ } (A \cdot ys) := \text{fby}_1 \text{ } v \text{ } xs \text{ } ys$$
$$\text{fby}'_1 \text{ } _ \text{ } xs \text{ } (\text{err} \cdot ys) := \text{map } (\lambda x. \text{err}) \text{ } xs$$
$$\text{fby}'_1 \text{ } _ \text{ } xs \text{ } (_ \cdot ys) := \text{map } (\lambda x. \text{err}_{\text{sync}}) \text{ } xs$$
$$\text{fby}_1 \text{ } v \text{ } (A \cdot xs) \text{ } ys := A \cdot \text{fby}'_1 (\text{Some } v) \text{ } xs \text{ } ys$$
$$\text{fby}_1 \text{ } v \text{ } (v_x \cdot xs) \text{ } ys := v \cdot \text{fby}'_1 \text{ None } xs \text{ } ys$$
$$\text{fby}_1 \text{ } v \text{ } (\text{err} \cdot xs) \text{ } ys := \text{err} \cdot \text{map } (\lambda x. \text{err}) \text{ } xs$$

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	...

$$\text{fby } (A \cdot xs) \text{ } ys := A \cdot \text{fby}_A \text{ } xs \text{ } ys$$

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

$$\text{fby } (\text{err} \cdot xs) \text{ } ys := \text{err} \cdot \text{map } (\lambda x. \text{err}) \text{ } xs$$

$$\text{fby}_A \text{ } xs \text{ } (A \cdot ys) := \text{fby } xs \text{ } ys$$

$$\text{fby}_A \text{ } xs \text{ } (\text{err} \cdot ys) := \text{map } (\lambda x. \text{err}) \text{ } xs$$

$$\text{fby}_A \text{ } xs \text{ } (v \cdot ys) := \text{map } (\lambda x. \text{err}_{\text{sync}}) \text{ } xs$$

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

$$\text{fby}'_1 (\text{Some } v) \text{ } xs \text{ } (A \cdot ys) := \text{fby}_1 \text{ } v \text{ } xs \text{ } ys$$


$$\text{fby}'_1 \text{ } _ \text{ } xs \text{ } (\text{err} \cdot ys) := \text{map } (\lambda x. \text{err}) \text{ } xs$$

$$\text{fby}'_1 \text{ } _ \text{ } xs \text{ } (_ \cdot ys) := \text{map } (\lambda x. \text{err}_{\text{sync}}) \text{ } xs$$

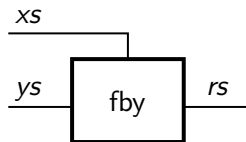
$$\text{fby}_1 \text{ } v \text{ } (A \cdot xs) \text{ } ys := A \cdot \text{fby}'_1 (\text{Some } v) \text{ } xs \text{ } ys$$

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

$$\text{fby}_1 \text{ } v \text{ } (\text{err} \cdot xs) \text{ } ys := \text{err} \cdot \text{map } (\lambda x. \text{err}) \text{ } xs$$

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

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	...

$$\text{fby } (A \cdot xs) \text{ } ys := A \cdot \text{fby}_A xs \text{ } ys$$

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

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

$$\text{fby}'_1 (\text{Some } v) \text{ } xs \text{ } (A \cdot ys) := \text{fby}_1 v \text{ } xs \text{ } ys$$

$$\text{fby } (\text{err} \cdot xs) \text{ } ys := \text{err} \cdot \text{map } (\lambda x. \text{err}) \text{ } xs$$

$$\text{fby}'_1 _ \text{ } xs \text{ } (\text{err} \cdot ys) := \text{map } (\lambda x. \text{err}) \text{ } xs$$

$$\text{fby}_A xs \text{ } (A \cdot ys) := \text{fby } xs \text{ } ys$$

$$\text{fby}'_1 _ \text{ } xs \text{ } (_ \cdot ys) := \text{map } (\lambda x. \text{err}_{\text{sync}}) \text{ } xs$$

$$\text{fby}_A xs \text{ } (\text{err} \cdot ys) := \text{map } (\lambda x. \text{err}) \text{ } xs$$

$$\text{fby}_1 v \text{ } (A \cdot xs) \text{ } ys := A \cdot \text{fby}'_1 (\text{Some } v) \text{ } xs \text{ } ys$$

$$\text{fby}_A xs \text{ } (v \cdot ys) := \text{map } (\lambda x. \text{err}_{\text{sync}}) \text{ } xs$$

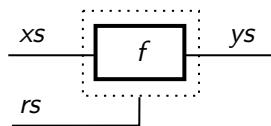
$$\text{fby}_1 v \text{ } (v_x \cdot xs) \text{ } ys := v \cdot \text{fby}'_1 \text{ None } xs \text{ } ys$$

$$\text{fby}_1 v \text{ } (\text{err} \cdot xs) \text{ } ys := \text{err} \cdot \text{map } (\lambda x. \text{err}) \text{ } xs$$

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

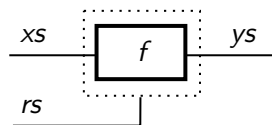
🔗 **Théorème** : et aussi, $\lfloor \text{fby } xs \text{ } ys \rfloor_A \preceq \text{fby}^\# \lfloor xs \rfloor_A \lfloor ys \rfloor_A$

Construction : réinitialisation modulaire



xs	1	1	1	A	1	1	1	1	...
rs	F	F	F	A	T	F	F	T	...
ys	1	2	3	A	1	2	3	1	...

Construction : réinitialisation modulaire

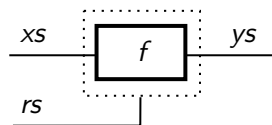


xs	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	\dots
rs	F	F	F	A	T	F	F	T	\dots
ys	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	\dots

[Bourke, Brun, Pouzet, 2020]

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

Construction : réinitialisation modulaire

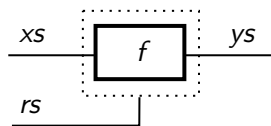


xs	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	\dots
rs	F	F	F	A	T	F	F	T	\dots
ys	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	\dots

[Bourke, Brun, Pouzet, 2020]

$$\begin{aligned} & f \vdash [x_1 \cdot x_2 \cdot x_3 \cdot x_4] \Downarrow [y_1 \cdot y_2 \cdot y_3 \cdot y_4] \\ \wedge & f \vdash [x_5 \cdot x_6 \cdot x_7] \Downarrow [y_5 \cdot y_6 \cdot y_7] \\ \wedge & f \vdash [x_8] \Downarrow [y_8] \\ \wedge & \dots \\ & (\text{masquage} : [x_2 \cdot x_3] \equiv A \cdot x_2 \cdot x_3 \cdot A \cdot A \dots) \end{aligned}$$

Construction : réinitialisation modulaire



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

[Hamon, Pouzet, 2001]

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

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

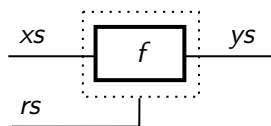
merge cs (f (when $cs\ xs$))

(reset_f (whenot $cs\ rs$) (whenot $cs\ xs$))

[Bourke, Brun, Pouzet, 2020]

$f \vdash [x_1 \cdot x_2 \cdot x_3 \cdot x_4] \Downarrow [y_1 \cdot y_2 \cdot y_3 \cdot y_4]$
 $\wedge f \vdash [x_5 \cdot x_6 \cdot x_7] \Downarrow [y_5 \cdot y_6 \cdot y_7]$
 $\wedge f \vdash [x_8] \Downarrow [y_8]$
 $\wedge \dots$
 (masquage : $[x_2 \cdot x_3] \equiv A \cdot x_2 \cdot x_3 \cdot A \cdot A \dots$)

Construction : réinitialisation modulaire



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

[Hamon, Pouzet, 2001]

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

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

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

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

[Gérard, 2013]

$\text{sreset}'_f rs xs$

$:= \text{sreset}'_f rs xs (f \text{ } xs)$

$\text{sreset}'_f (T \cdot rs) xs ys$

$\simeq \text{sreset}'_f (F \cdot rs) xs (f \text{ } xs)$

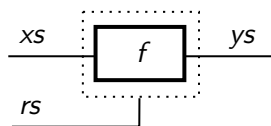
$\text{sreset}'_f (F \cdot rs) (x \cdot xs) (y \cdot ys)$

$\simeq y \cdot (\text{sreset}'_f rs xs ys)$

$\text{sreset}'_f (A \cdot rs) (x \cdot xs) (y \cdot ys)$

$\simeq y \cdot (\text{sreset}'_f rs xs ys)$

Construction : réinitialisation modulaire



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

[Hamon, Pouzet, 2001]

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

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

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

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

[Gérard, 2013]

$\text{sreset}'_f rs xs$

$:= \text{sreset}'_f rs xs (f \text{ } xs)$

$\text{sreset}'_f (T \cdot rs) xs ys$

$\simeq \text{sreset}'_f (F \cdot rs) xs (f \text{ } xs)$

$\text{sreset}'_f (F \cdot rs) (x \cdot xs) (y \cdot ys)$

$\simeq y \cdot (\text{sreset}'_f rs xs ys)$

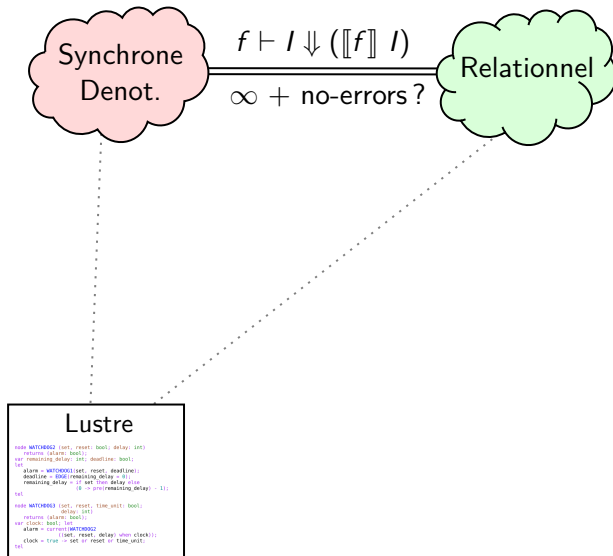
$\text{sreset}'_f (A \cdot rs) (x \cdot xs) (y \cdot ys)$

$\simeq y \cdot (\text{sreset}'_f rs xs ys)$

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

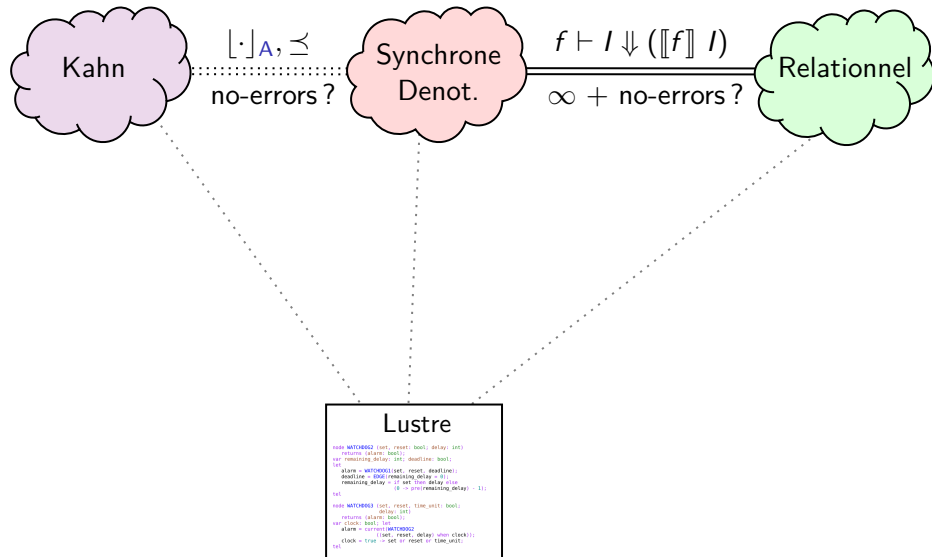
Approche proposée

Traitement des erreurs



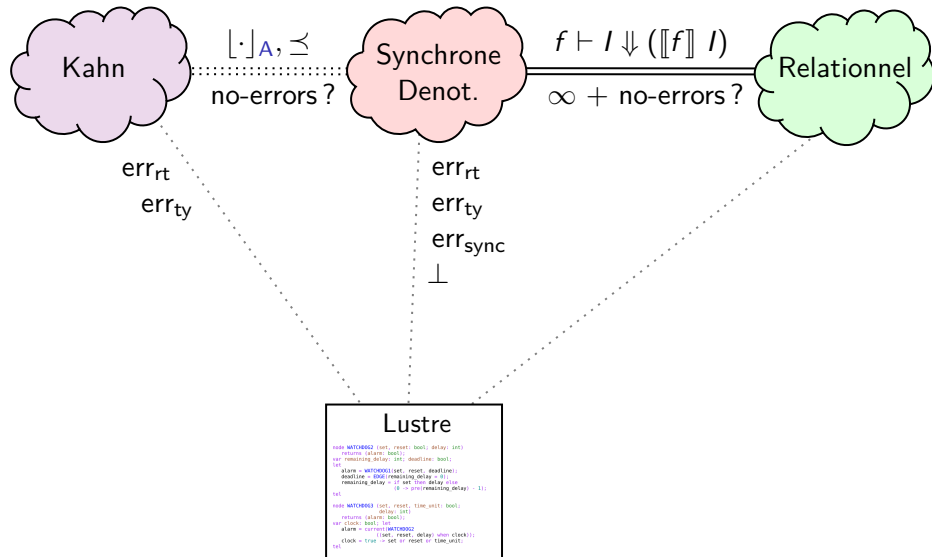
Approche proposée

Traitement des erreurs



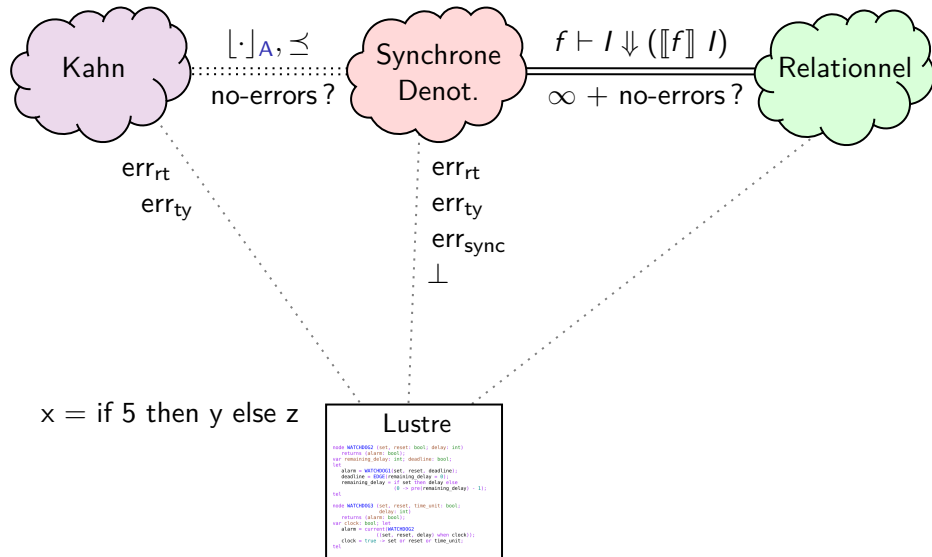
Approche proposée

Traitement des erreurs



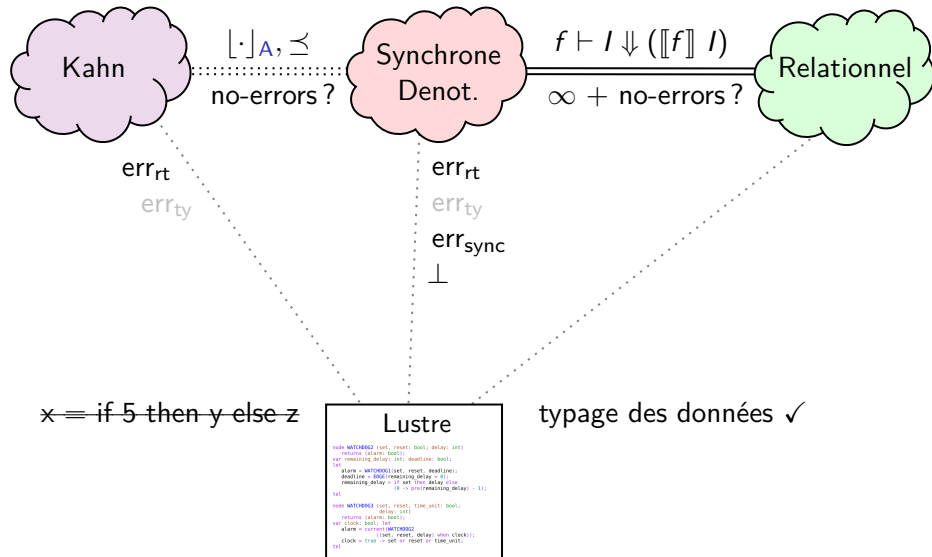
Approche proposée

Traitement des erreurs



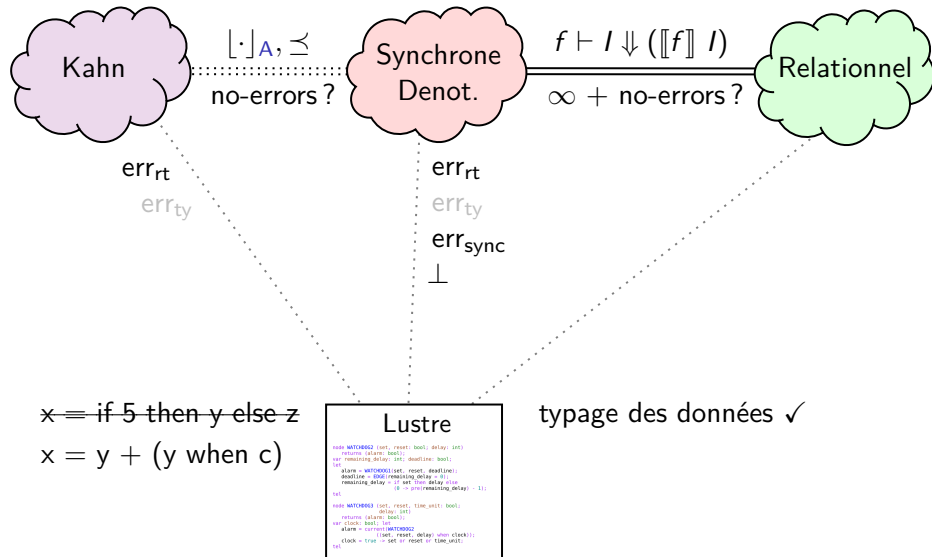
Approche proposée

Traitement des erreurs



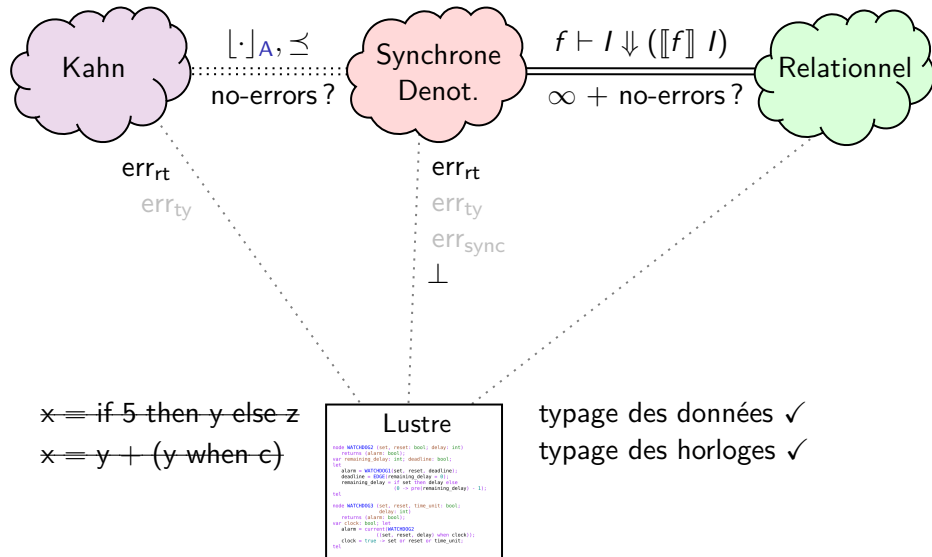
Approche proposée

Traitement des erreurs



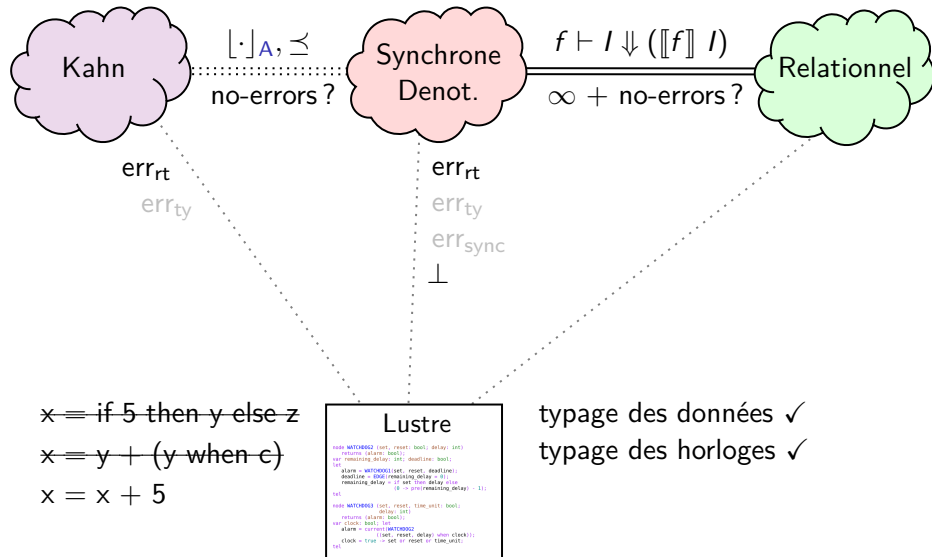
Approche proposée

Traitement des erreurs



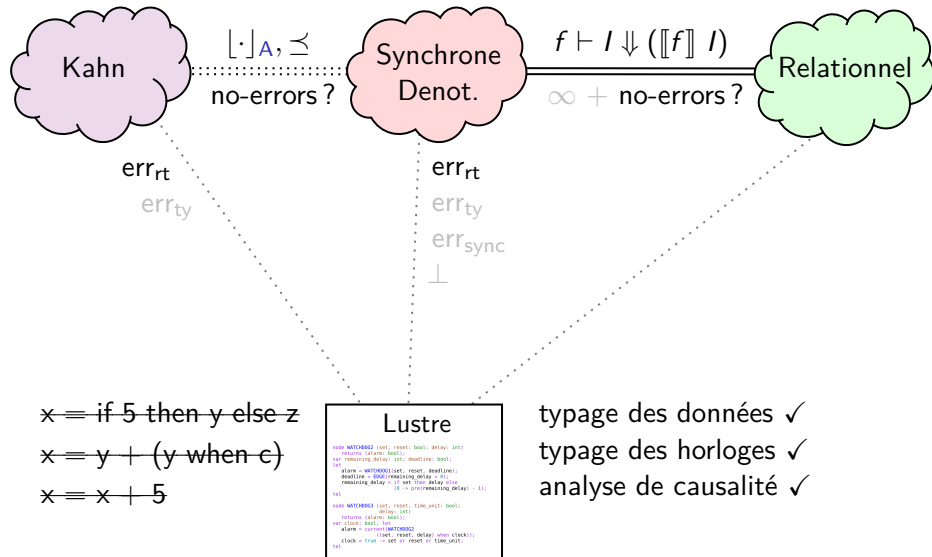
Approche proposée

Traitement des erreurs



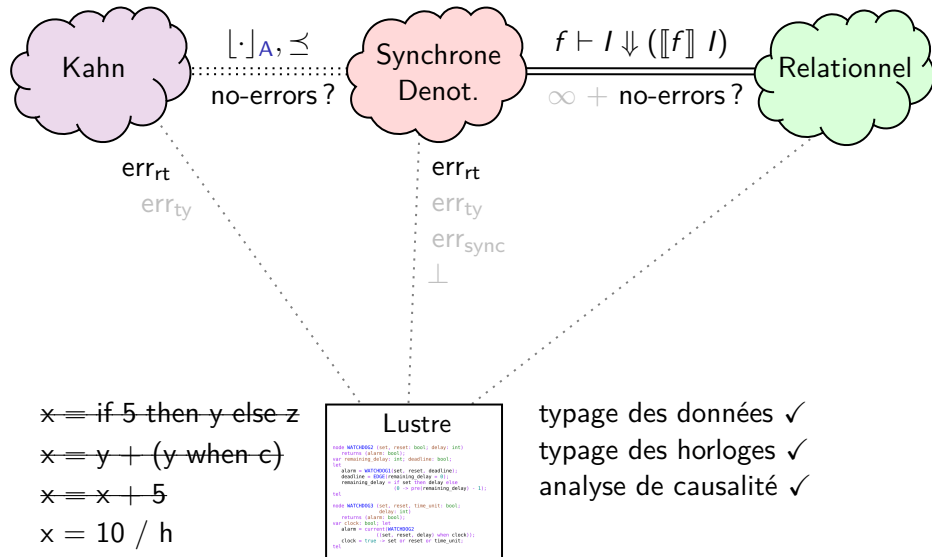
Approche proposée

Traitement des erreurs



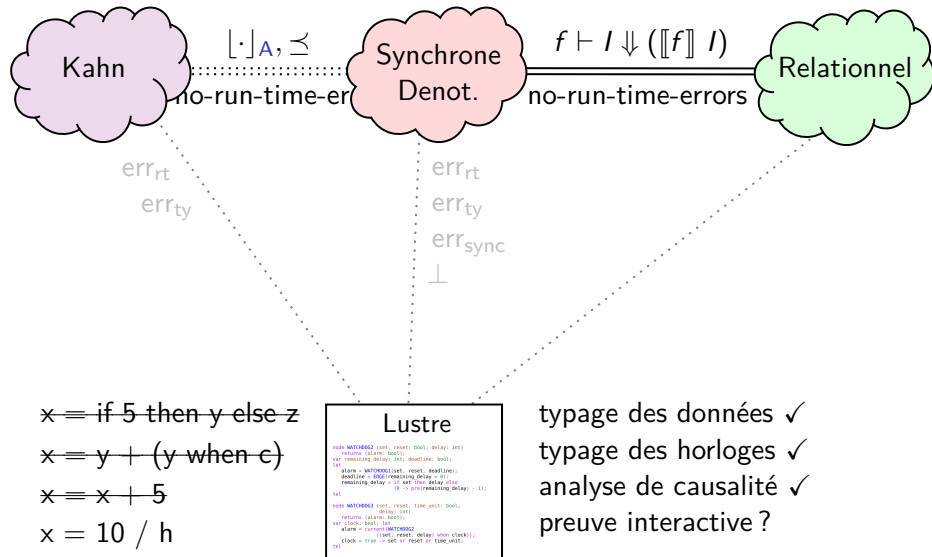
Approche proposée

Traitement des erreurs



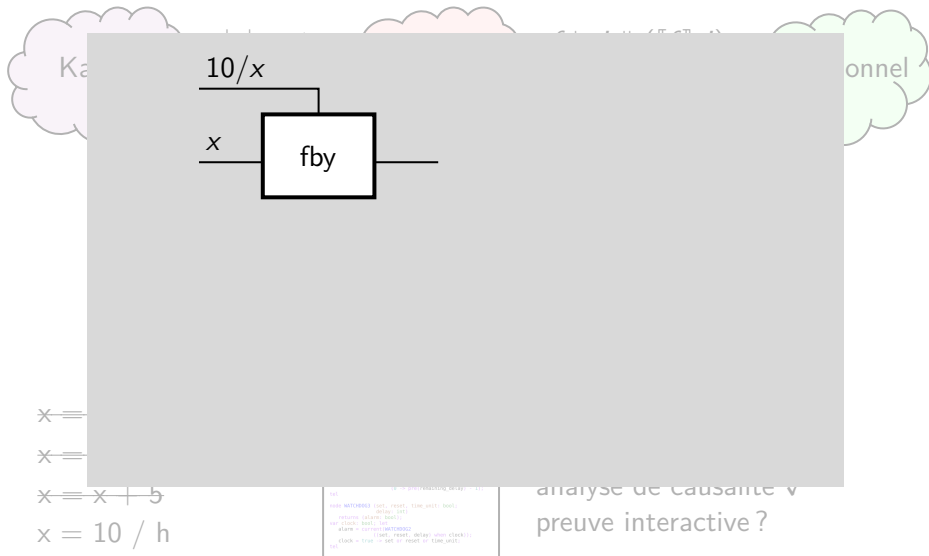
Approche proposée

Traitement des erreurs



Approche proposée

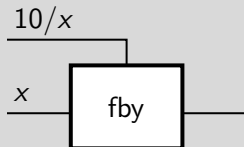
Traitement des erreurs



analyse de causale v
preuve interactive ?

Approche proposée

Traitement des erreurs



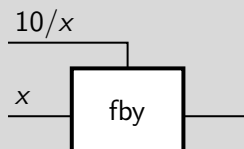
x	5	2	1	0	2	5	...
fby#	2	5	2	1	0	2	...

```
let  
  state WATCHDOG (set, reset, time_unit: bool;  
    delay: int)  
  returns (alarm: bool);  
var clock: bool; let  
  alarm = current(WATCHDOG2)  
  clock = true = set or reset or time_unit;  
res;
```

analyse de causalité v
preuve interactive?

Approche proposée

Traitement des erreurs



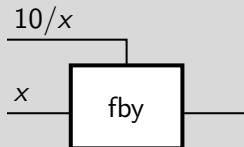
x	5	2	1	0	2	5	...
fby#	2	5	2	1	0	2	...
fby	2	5	2	err _{rt}	err _{rt}	err _{rt}	...

```
let  
  state WATCHDOG (set, reset, time_unit: bool;  
    delay: int)  
  returns (alarm: bool);  
var clock: bool; let  
  alarm = current(WATCHDOG);  
  clock = true => set or reset or time_unit;  
res;
```

analyse de causalité v
preuve interactive?

Approche proposée

Traitement des erreurs



Autres obligations ?

Variations sur fby[#] ?

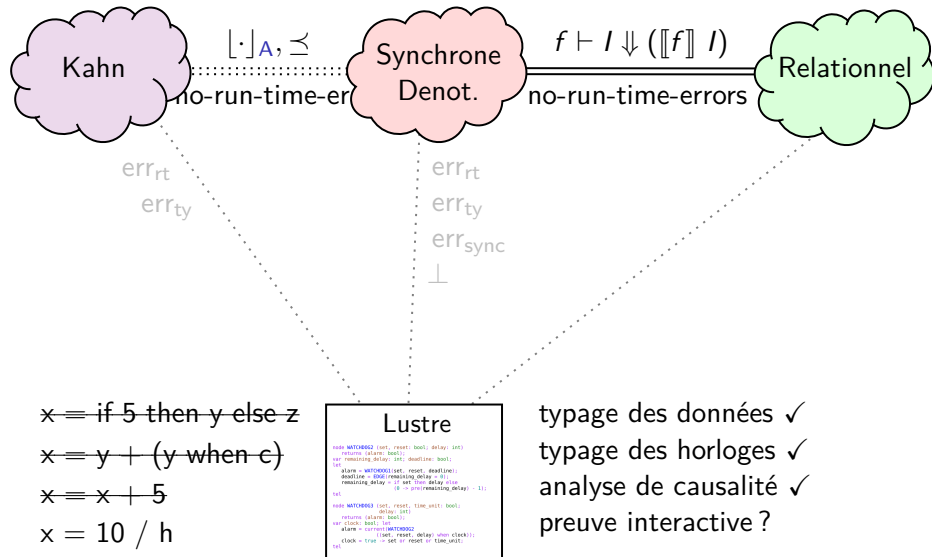
x	5	2	1	0	2	5	...
fby [#]	2	5	2	1	0	2	...
fby	2	5	2	err _{rt}	err _{rt}	err _{rt}	...

```
let
  is = previousDelay == 1;
  note WATCHDOG (set, reset, timeUnit: bool;
    delay: set;
    returns: alarm: bool);
  var clock: bool; let
    alarm = current(WATCHDOG);
    clock = true == set or reset or timeUnit;
  res;
```

analyse de causalité v
preuve interactive ?

Approche proposée

Traitement des erreurs



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)

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

Analyse statique

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

$n/0$	$n\%0$	$n \gg 64$	$n \ll 64$
$\text{MIN_INT} / -1$	$\text{MIN_INT} \% -1$		
$(\text{int})\text{NaN}$	$(\text{int})\text{Infty}$	$(\text{int})2^{63}-1.0$	$(\text{int})-2^{63}.0$

Procédure : interdire ces opérations lorsque l'opérande est inconnue

- ▶ $\text{check-ops}(4 \% x) = \text{F}$
- ▶ $\text{check-ops}(x / 2) = \text{T}$
- ▶ ...

Analyse statique

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

$n/0$	$n\%0$	$n \gg 64$	$n \ll 64$
$\text{MIN_INT} / -1$	$\text{MIN_INT} \% -1$		
$(\text{int})\text{NaN}$	$(\text{int})\text{Infty}$	$(\text{int})2^{63}-1.0$	$(\text{int})-2^{63}.0$

Procédure : interdire ces opérations lorsque l'opérande est inconnue

- ▶ $\text{check-ops}(4 \% x) = F$
- ▶ $\text{check-ops}(x / 2) = T$
- ▶ ...

🐞 **Théorème** : $\forall x, f$, si $\text{check-ops}(f) = T$ alors $\text{no-run-time-errors } f \ x$

Conclusion

Une correction de la compilation renforcée

Théorème (avant¹)

si compile $f = \text{OK } asm$

et welltyped-inputs $f \ xs$

et $f \vdash xs \Downarrow ys$

alors $asm \Downarrow \langle \text{Load } (xs(i)) \cdot \text{Store } (ys(i)) \rangle_{i=0}^{\infty}$

1. avec machines à états

Conclusion

Une correction de la compilation renforcée

Théorème (avant¹)

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

Théorème (après²)

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

1. avec machines à états

2. sans machines à états

Conclusion

Une correction de la compilation renforcée

Théorème (avant¹)

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

Théorème (après²)

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

-
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