Esterel de A à Z

2. Sémantiques, causalité et constructivité des langages synchrones

Gérard Berry

Collège de France
Chaire Algorithmes, machines et langages

Cours 3 du 14 février 2018
suivi du séminaire ReactiveML de Louis Mandel

gerard.berry@college-de-france.fr
http://www-sop.inria.fr/members/Gerard.Berry

COLLÈGE
DE FRANCE
1530

Le synchronisme parfait

- Exécution par une suite d'instants discrets
  - réactions à des entrées en produisant des sorties
- Administration infiniment rapide
  - transmettre le contrôle ne prend aucun temps
  - seul pause prend du temps, ne rendant le contrôle qu'à l'instant suivant
- Communication infiniment rapide
  - émettre un signal ne prend aucun temps
  - le signal est reçu dès qu'il est émis par ceux qui l'écoutent

Ces principes simplifient la programmation
mais ni la sémantique ni la compilation !
$\Longrightarrow$ comprendre les problèmes de causalité

Les questions posées par Esterel

- Comment donner un sens précis aux programmes Esterel tout en préservant leur intuition ?

⇒ sémantique formelle (mathématique)

- La communication synchrone conduit rapidement à des paradoxes. Que traduisent-ils, et comment les résoudre ?

⇒ étude des cycles de causalité, nous conduisant de la logique classique à la logique intuitionniste

- Comment faire passer tout ça à la pratique ?

⇒ compilation, optimisation et debugging entièrement fondés sur la sémantique formelle

Agenda

1. Quel type de sémantique choisir ?

Comment définir la sémantique d'un langage?

1. Par un mode d'emploi textuel décrivant ce que font les instructions et comment elles s'enchaînent
   - la méthode usuelle, indispensable, mais souvent floue
   - pas de critère de cohérence ni de complétude !

2. Mieux : utiliser un modèle mathématique extrinsèque, indépendant du langage précis
   - sémantique opérationnelle : ex. traduction sur machine abstraite
   - sémantique par réécriture de termes (Kleene)
   - sémantique logique : respect d'assertions logiques (Turing, …)
   - sémantique dénotationnelle : fonctions, ordres, catégories, etc.
   - SOS : sémantique opérationnelle structurelle (G. Plotkin)

Indispensables pour savoir ce que font les programmes
Interactions subtiles entre ces approches

Sémantique dénotationnelle (Scott − Strachey)

- Sémantique de Scott par ordres partiels (1970)

  fact (n) = if n=0 then 1 else n ∗ fact (n−1)

  voir ce programme comme une équation fonctionnelle
  bouclage possible → fonctions partielles, traitement de l'indéfini, ordre d'information, catégories de domaines et fonctions, etc. (Voir cours 2010)

- Applications pratiques
  - (méthode VDM de définition des langages)
  - sémantique flots de données pour Kahn / Lustre / Signal
  - interprétation abstraite (Cousot), voir cours du 11/03/2015

    SCADE : non-débordement de pile, absence d'erreurs run-time
    garanties de temps d'exécution, etc.
    → industrialisé par www.absint.com

Mais mal adaptée à la sémantique du parallélisme et de la préemption impératifs en Esterel

Sémantique par réécriture

- Kleene (195?) : équations récursives comme modèle de la calculabilité
  - variables : x, y, x, …
  - symboles de fonction : Z, S, P, Add, Mult, Fact,…
  - équations entres termes

    P(S(x)) = x
    Fact(Z) = S(Z),
    Fact(S(x)) = Mult(S(x), Fact(x))
  - réécritures : Fact(S(S(S(Z)))) = S(S(S(S(S(Z)))))

  → mécanisée dans les prouveurs ACL2, Agda, (Coq)

Sémantique toute récente d'Esterel
par Spencer Florence et Robby Findler,
Northwestern University Chicago

Assertions logiques

- Pas utilisées pour la sémantique, mais indispensables pour la vérification formelle des programmes

```
module Runner :
input Morning, Second , Meter , Step , Lap ;
output RunSlowly , Jump , Breathe , RunFast ;
   every Morning do
      abort
         loop
            abort sustain RunSlowly when 100 Meter ;
            abort
               every Step do emit { Jump , Breathe } end every
            when 15 Second ;
            sustain RunFast
         each Lap
      when 4 Lap
   end every
 ||
   sustain assert AtMostOneBehavior = RunSlowly # Jump # RunFast
end module
```

Sémantiques SOS, les références

- Invention majeure de Gordon Plotkin (1983)
  décrire un langage comme un système logique,
  et les calculs comme des preuves dans ce système
- Préfigurée par les calculs de processus de Milner

  CCS = Calculus of Communicating Systems (1984)
  voir cours du
  SCCS = Synchronous CCS → Meije de G. Boudol

  pas assez puissantes pour le concept central d'Esterel

Faire les choses en même temps,
mais dans le bon ordre !

Agenda

1. Quelle type de sémantique choisir ?
2. Rappel : le noyau d'Esterel Pur

`nothing`	*ne fait rien et termine instantanément*
`emit` S	*émet le signal* S
`pause`	*arrête l'exécution,* *puis redémarre au prochain instant*
p `;` q	*exécute* p *puis* q *si et quand* p *termine*
`loop` p `end`	*répète* p *indéfiniment*
p `||` q	*exécute* p *et* q *en parallèle synchrone*
`trap` T `in` p `end`	*déclare et gère la trappe* T *dans* p
`exit` T^k	*lève la trappe* T *d'index* k
`if` S `then` p `else` q `end`	*exécute* p *si* S *est présent,* q *sinon*
`suspend` S `when` p	*lance* p *puis le suspend si* S *est présent* *termine si* p *termine* *et premier instant ou* S *absent*
`signal` S `in` p `end`	*déclare le signal* S *local dans* p

Agenda

1. Quelle type de sémantique choisir ?
2. Rappel : le noyau d'Esterel Pur
3. Codes de retour : terminaison, exits

Codes de retour numériques (Gonthier)

- La clef de la propagation du contrôle
- Trois types de fin d'exécution d'une instruction
  - terminaison normale : code de retour 0
  - atteinte d'une pause : code 1
  - sortie de la trappe la plus proche : 2
    2e plus proche : 3
    …

Si les codes de retour de p et q sont k et k'
alors le code de retour de p||q est max(k,k')

Mumérotation des exit

```
input I, J, K;
output X, Y;
trap T in
   trap U in
      if I then pause end
   ||
      if J then exit U2 end
   ||
      if K then exit T3 end
   handle U do emit X
   end trap;
handle T do emit Y
end trap
```

Mumérotation des exit (1)

```
input I, J, K;
output X, Y;
trap T in
   trap U in
      if I then pause end
   ||
      if J then exit U2 end
   ||
      if K then exit T3 end
   handle U do emit X
   end trap;
handle T do emit Y
end trap
```

I J K → 0 0 0 → 0 →

Mumérotation des exit (2)

```
input I, J, K;
output X, Y;
trap T in
   trap U in
      if I then pause end
   ||
      if J then exit U2 end
   ||
      if K then exit T3 end
   handle U do emit X
   end trap;
handle T do emit Y
end trap
```

I J K → 0 0 0 → 0 →

I J K → 1 ~~0 0~~ → 1 →

Mumérotation des exit (3)

```
input I, J, K;
output X, Y;
trap T in
   trap U in
      if I then pause end
   ||
      if J then exit U2 end
   ||
      if K then exit T3 end
   handle U do emit X
   end trap;
handle T do emit Y
end trap
```

I J K → 0 0 0 → 0 →

I J K → 1 ~~0~~ ~~0~~ → 1 →

I J K → ~~1~~ 2 ~~0~~ → 2 → X

Mumérotation des exit (4)

```
input I, J, Y ;
output X, Y ;
trap T in
   trap U in
      if I then pause end
   ||
      if J then exit U2 end
   ||
      if K then exit T3 end
   handle U do emit X
   end trap ;
handle T do emit Y
end trap
```

I J K → 0 0 0 → 0 →

I J K → 1 ~~0~~ ~~0~~ → 1 →

I J K → ~~X~~ 2 ~~0~~ → 2 → X

I J K → ~~0~~ ~~0~~ 3 → 3 → Y

Mumérotation des exit (5)

```
input I, J, K;
output X, Y;
trap T in
   trap U in
      if I then pause end
   ||
      if J then exit U2 end
   ||
      if K then exit T3 end
   handle U do emit X
   end trap;
handle T do emit Y
end trap
```

I J K → 0 0 0 → 0 →

I J K → 1 ~~0 0~~ → 1 →

I J K → ~~X~~ 2 ~~0~~ → 2 → X

I J K → ~~0 0~~ 3 → 3 → Y

I J K → ~~0 X~~ 3 → 3 → Y

Mumérotation des exit (6)

```
input I, J, K;
output X, Y;
trap T in
   trap U in
      if I then pause end
   ||
      if J then exit U2 end
   ||
      if K then exit T3 end
   handle U do emit X
   end trap;
handle T do emit Y
end trap
```

I J K → 0 0 0 → 0 →

I J K → 1 ~~0~~ ~~0~~ → 1 →

I J K → ~~1~~ 2 ~~0~~ → 2 → X

I J K → ~~0~~ ~~0~~ 3 → 3 → Y

I J K → ~~0~~ ~~0~~ 3 → 3 → Y

I J K → ~~1~~ ~~2~~ 3 → 3 → Y

Deux auxiliaires techniques

- Sortie de trappe : $k \rightarrow \downarrow k$

  $\downarrow 0 = 0$

  $\downarrow 1 = 1$

  $\downarrow 2 = 0$ la trappe courante termine

  $\downarrow k = k-1$ si $k \geq 3$ l'exit est propagé à la trappe supérieure

- Nettoyage des termes morts : $\delta^k(p)$

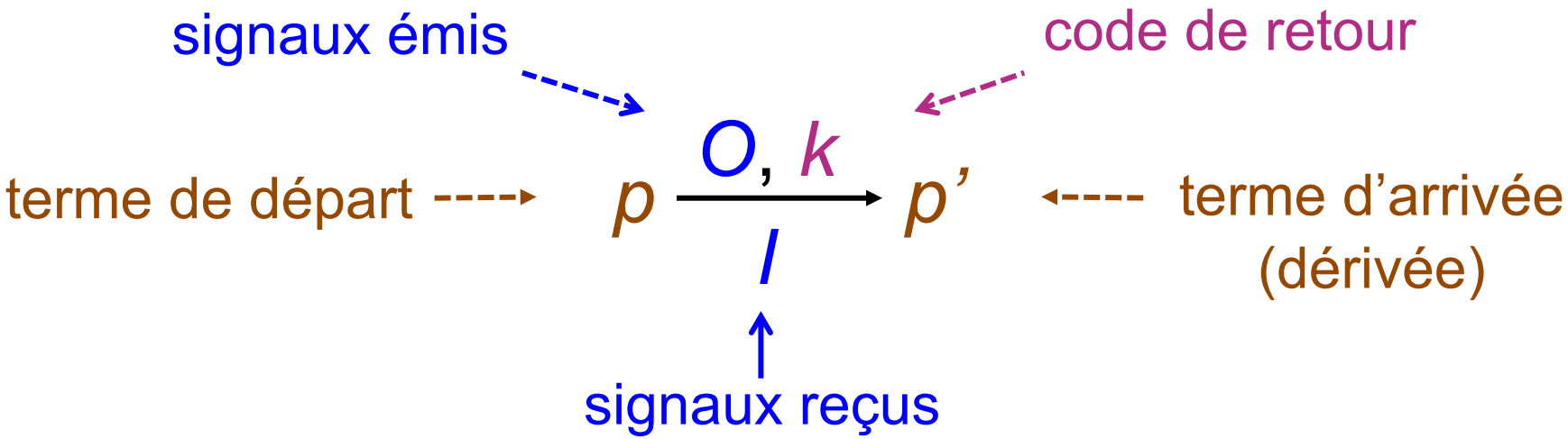
  $\delta^k(p) = p$ si $k=1$

  $\delta^k(p) =$ nothing sinon

Agenda

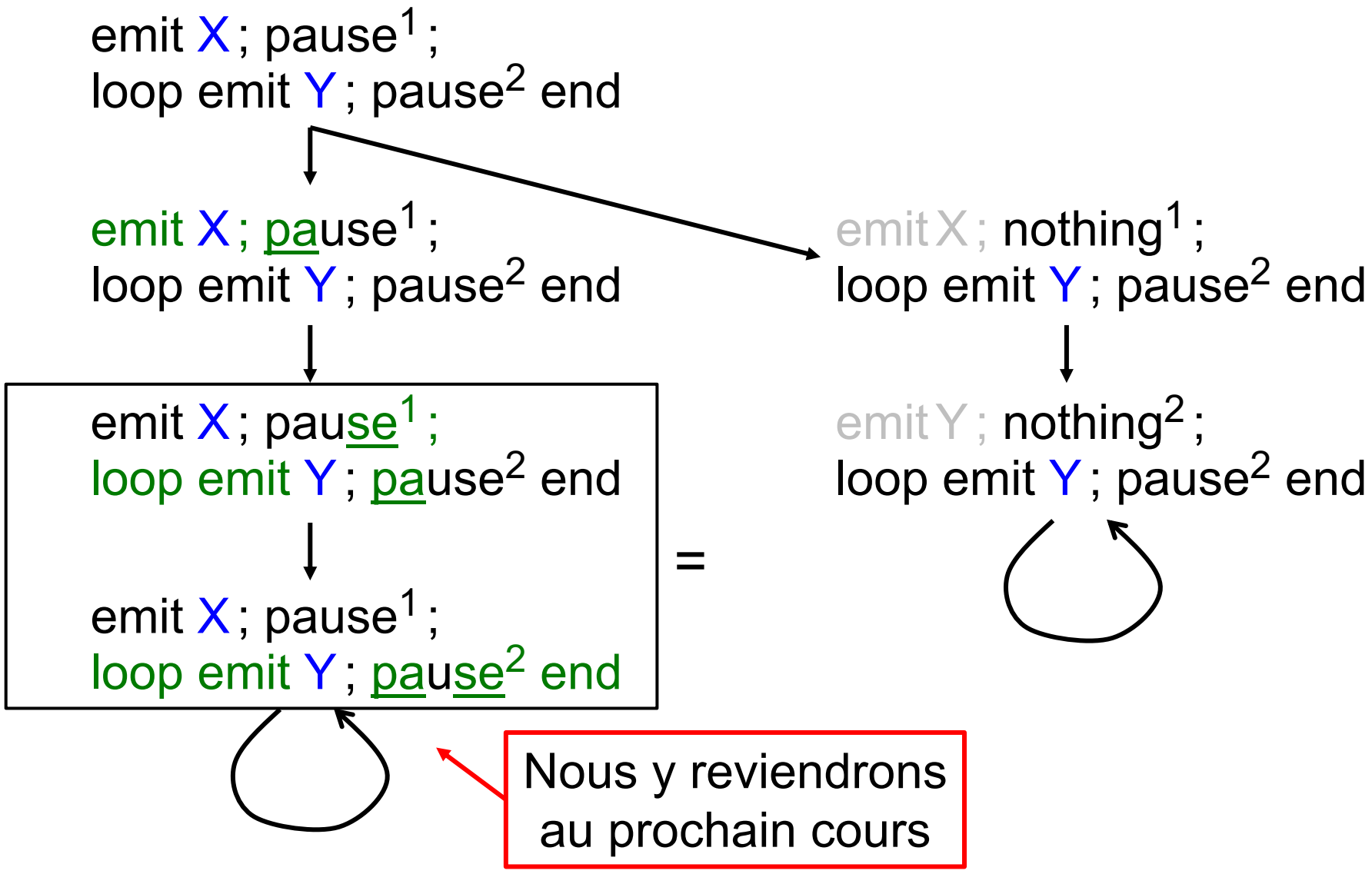
1. Quelle type de sémantique choisir ?
2. Rappel : le noyau d'Esterel Pur
3. Codes de retour : terminaison, exits
4. La sémantique comportementale (Berry − Cosserat, 1984)

La sémantique comportementale (SOS)



Une seule transition du programme par réaction,
calculée récursivement en fonction
de celles de ses sous-termes

Equivalence décoration / transition



Les règles comportementales

$$\texttt{nothing} \xrightarrow[I]{\emptyset,0} \texttt{nothing} \qquad \text{(nothing)}$$

$$\texttt{pause} \xrightarrow[I]{\emptyset,1} \texttt{nothing} \qquad \text{(pause)}$$

$$\texttt{exit}\ T^k \xrightarrow[I]{\emptyset,k} \texttt{nothing} \qquad \text{(exit)}$$

$$\texttt{emit}\ S \xrightarrow[I]{\{S\},0} \texttt{nothing} \qquad \text{(emit)}$$

$$\frac{S \in I \quad p \xrightarrow[I]{O,k} p'}{\texttt{if}\ S\ \texttt{then}\ p\ \texttt{else}\ q\ \texttt{end} \xrightarrow[I]{O,k} p'} \qquad \text{(présent)}$$

$$\frac{S \notin I \quad q \xrightarrow[I]{O,k} q'}{\texttt{if}\ S\ \texttt{then}\ p\ \texttt{else}\ q\ \texttt{end} \xrightarrow[I]{O,k} q'} \qquad \text{(absent)}$$

$$\frac{p \xrightarrow[I]{O,k} p'}{\texttt{suspend}\ S\ \texttt{when}\ p \xrightarrow[I]{O,k} \delta^k(\texttt{suspimm}\ S\ \texttt{when}\ p')} \qquad \text{(suspend)}$$

$$\frac{p \xrightarrow[I]{O,0} p' \quad q \xrightarrow[I]{O',k} q'}{p;q \xrightarrow[I]{O\cup O',k} q'} \qquad \text{(séquence-0)}$$

$$\frac{p \xrightarrow[I]{O,k} p' \quad k\neq 0}{p;q \xrightarrow[I]{O,k} \delta^k(p';q)} \qquad \text{(séquence-k)}$$

$$\frac{p \xrightarrow[I]{O,k} p' \quad k\neq 0}{\texttt{loop}\ p\ \texttt{end} \xrightarrow[I]{O,k} \delta^k(p';\texttt{loop}\ p\ \texttt{end})} \qquad \text{(loop)}$$

$$\frac{p \xrightarrow[I]{O,k} p' \quad q \xrightarrow[I]{O',l} q' \quad m = \max(k,l)}{p \parallel q \xrightarrow[I]{O\cup O',m} \delta^m(p' \parallel q')} \qquad \text{(parallèle)}$$

$$\frac{p \xrightarrow[I]{O,k} p'}{\texttt{trap}\ T\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O,\downarrow k} \delta^{\downarrow k}(\texttt{trap}\ T\ \texttt{in}\ p'\ \texttt{end})} \qquad \text{(trap)}$$

$$\frac{p \xrightarrow[I\cup\{S\}]{O,k} p' \quad S \in O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O\setminus\{S\},k} \delta^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})} \qquad \text{(signal+)}$$

$$\frac{p \xrightarrow[I\setminus\{S\}]{O,k} p' \quad S \notin O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O,k} \delta^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})} \qquad \text{(signal−)}$$

14 règles individuellement simples pour les 11 primitives d'Esterel.

Tout est dans leur interaction !

1. Règles de base

$$\texttt{nothing} \xrightarrow[I]{\emptyset, 0} \texttt{nothing} \qquad \text{(nothing)}$$

$$\texttt{pause} \xrightarrow[I]{\emptyset, 1} \texttt{nothing} \qquad \text{(pause)}$$

$$\texttt{exit}\ T^k \xrightarrow[I]{\emptyset, k} \texttt{nothing} \qquad \text{(exit)}$$

$$\texttt{emit}\ S \xrightarrow[I]{\{S\}, 0} \texttt{nothing} \qquad \text{(emit)}$$

2. Test de signal et suspension

$$\frac{S \in I \qquad p \xrightarrow[I]{O,k} p'}{\texttt{if } S \texttt{ then } p \texttt{ else } q \texttt{ end} \xrightarrow[I]{O,k} p'} \text{(présent)}$$

$$\frac{S \notin I \qquad q \xrightarrow[I]{O,k} q'}{\texttt{if } S \texttt{ then } p \texttt{ else } q \texttt{ end} \xrightarrow[I]{O,k} q'} \text{(absent)}$$

$$\frac{p \xrightarrow[I]{O,k} p'}{\texttt{suspend } S \texttt{ when } p \xrightarrow[I]{O,k} \delta^k(\texttt{suspimm } S \texttt{ when } p')} \text{(suspend)}$$

avec suspim p when S = await not S ; suspend p when S

3. séquence

$$\frac{p \xrightarrow[I]{O,0} p' \qquad q \xrightarrow[I]{O',k} q'}{p;q \xrightarrow[I]{O\cup O',k} q'} \quad \text{(séquence-0)}$$

$$\frac{p \xrightarrow[I]{O,k} p' \qquad k\neq 0}{p;q \xrightarrow[I]{O,k} \delta^k(p';q)} \quad \text{(séquence-k>0)}$$

Synchronisme : dans (séquence-0), transmission du contrôle dans la même transition, et union des signaux émis par *p* et *p'*

4. boucle

$$\frac{p \xrightarrow[I]{O,\,k} p' \qquad k \neq 0}{\texttt{loop } p \texttt{ end} \xrightarrow[I]{O,\,k} \delta^k(p' ; \texttt{loop } p \texttt{ end})} \text{(boucle)}$$

Attention : les boucles instantanées ($k=0$)
ne donnent lieu à aucune transition
(alternative, Tardieu : pour $k=0$, retourner ∞)

4. Parallèle et trappe

$$\frac{p \xrightarrow[I]{O,k} p' \qquad q \xrightarrow[I]{O',l} q' \qquad m = \max(k,l)}{p \parallel q \xrightarrow[I]{O \cup O',m} \delta^m(p' \parallel q')} \quad \text{(parallèle)}$$

$$\frac{p \xrightarrow[I]{O,k} p'}{\texttt{trap } T \texttt{ in } p \texttt{ end} \xrightarrow[I]{O,\downarrow k} \delta^{\downarrow k}(\texttt{trap } T \texttt{ in } p' \texttt{ end})} \quad \text{(trappe)}$$

Le parallèle synchronise la terminaison, la pause et les trappes
Il meurt dès qu'une branche fait exit d'une trappe externe

5. signal local

$$\frac{p \xrightarrow[I\cup\{S\}]{O,k} p' \qquad S \in O}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O\setminus\{S\},k} \delta^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})} \quad (\text{signal}+)$$

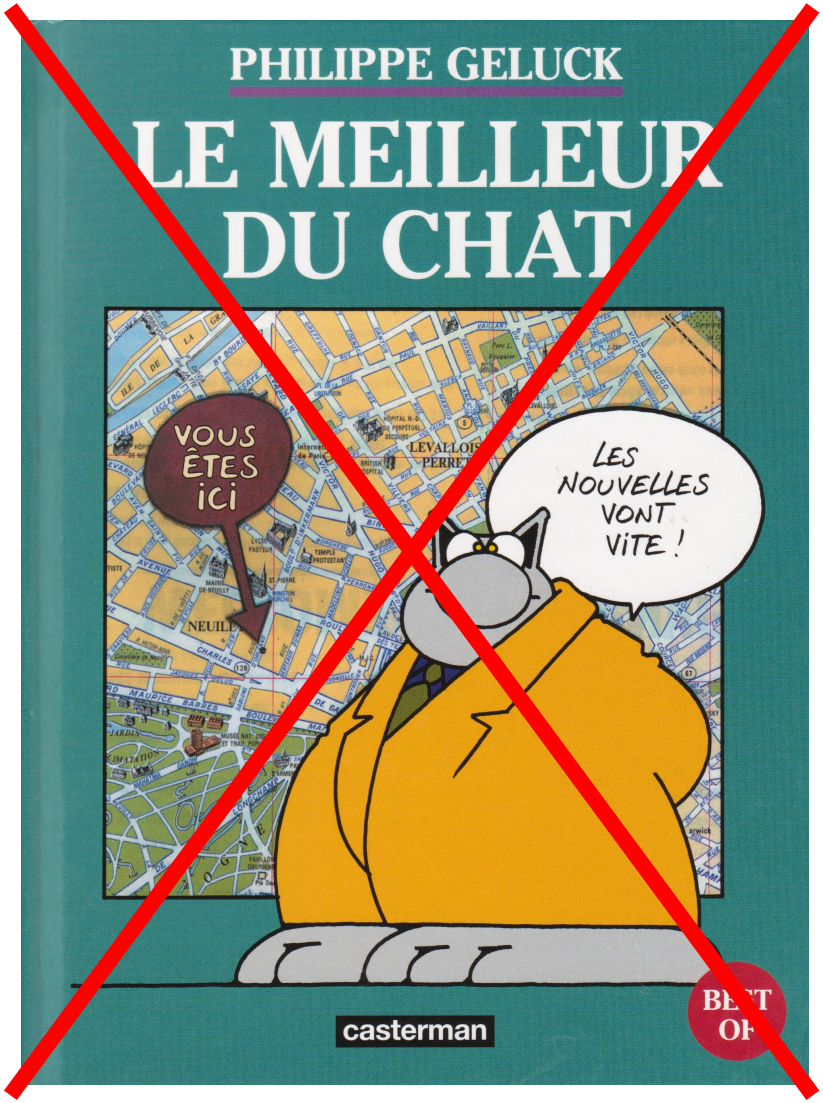
$$\frac{p \xrightarrow[I\setminus\{S\}]{O,k} p' \qquad S \notin O}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O,k} \delta^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})} \quad (\text{signal}-)$$

Deux règles imposant la consistance logique :
S est présent si et seulement s'il est émis
S est absent si et seulement s'il n'est pas émis

Agenda

1. Quelle type de sémantique choisir ?
2. Rappel : le noyau d'Esterel Pur
3. Codes de retour : terminaison, exits
4. La sémantique comportementale (Berry − Cosserat, 1984)
5. Problèmes de causalité

Communication instantanée et causalité selon le Chat de Philippe Geluck



après, mais en même temps !

Problèmes de non-sens

Esterel : if S then nothing else emit S end

Lustre : S = not S

Logique : pas de solution booléenne pour S

Esterel, sémantique comportementale :

pas de solution : ni (signal+) ni (signal−) ne va

Lustre : rejet car dépendance instantanée de S sur S
(testable par tri topologique)

Problème de non-déterminisme

Esterel : if S then emit S end

Lustre : S = S

Logique : S indifféremment vrai ou faux

Esterel, sémantique comportementale :

non-déterminisme

OK : (signal+) avec S∈I, S∈O

OK : (signal−) avec S∉I, S∉O

Lustre : rejet car dépendance instantanée de S sur S

(testable par tri topologique)

Non-déterminisme à 2

Esterel : if S1 else emit S2 end
|| if S2 else emit S1 end

Lustre : S1 = not S2
S2 = not S1

Logique : deux solutions, S1 et S2 opposés

Esterel, sémantique comportementale :

non-déterminisme comme en logique

OK : (signal+) avec S1∈I,O *et* (signal−) avec S2∉I,O

OK : (signal−) avec S1∉I,O *et* (signal+) avec S2∈I,O

Lustre : rejet car dépendance instantanée de S sur S
(testable par tri topologique)

Règles assurant le déterminisme (Tardieu)

$$\frac{p \xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \quad S \in O^- \qquad p \xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \quad S \in O^+}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O^+\setminus\{S\},k^+} \delta^{k^+}(\texttt{signal } S \texttt{ in } p^+ \texttt{ end})} \text{(signal++)}$$

$$\frac{p \xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \quad S \notin O^- \qquad p \xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \quad S \notin O^+}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O^-,k^-} \delta^{k^-}(\texttt{signal } S \texttt{ in } p^- \texttt{ end})} \text{(signal−−)}$$

Meilleures règles pour « signal S in p end » :
S présent (resp. absent) si et seulement si il est émis (resp. pas émis) indépendamment de sa présence dans I

Abort et la causalité

- Crime : « abort p when S » tué par S
- Suicide : p peut-il se donner la mort spontanément ?

```
signal Suicide in
   abort … emit Suicide … when Suicide
end
```

Impossible, car p n'est pas exécuté
si Suicide est présent !
Erreur de causalité

Weak abort et la causalité

- Crime : « weak abort p when S » tué par S
- Suicide : p peut-il se donner la mort spontanément ?

```
signal Suicide in
   weak abort … emit Suicide … when Suicide
end
```

Pas de problème, car préemption faible
(on peut remplacer signal par trap et emit par exit)

Le vrai danger : déterminisme suspect

Esterel : if S then emit S else emit S end

Lustre : S = S or not S

Hamlet : ToBe = ToBe or not ToBe

Logique : une seule solution, S vrai

Sémantique déterministe d'Esterel : S présent

seule règle applicable : (signal+) ou (signal++) pour S∈O

Lustre : rejet car dépendance instantanée de S sur S
(testable par tri topologique)

S présent est la seule solution en Esterel
mais cette solution logique n'est pas constructive !

Agenda

1. Quelle type de sémantique choisir ?
2. Rappel : le noyau d'Esterel Pur
3. Codes de retour : terminaison, exits
4. La sémantique comportementale (Berry − Cosserat, 1984)
5. Problèmes de causalité
6. Les bons cycles de causalité !

Mais pourquoi accepter les cycles ?

- Parce qu'ils peuvent être visiblement bons

```
if S then
   if S1 then emit S2 end
else
  if S2 then emit S1 end
end
```

la valeur de S
supprime une dépendance

```
if S1 then emit S2 end;
pause;
if S2 then emit S1 end
```

le changement d'instant
supprime une dépendance

Bien moins visible en Lustre !

```
if S then
   if S1 then emit S2 end
else
  if S2 then emit S1 end
end
```

→

```
S2 = S and S1
S1 = (not S) and S2
```

```
if S1 then emit S2 end;
pause;
if S2 then emit S1 end
```

→

```
  P = true → false
S2 = P and S1
S1 = (not P) and S2
```

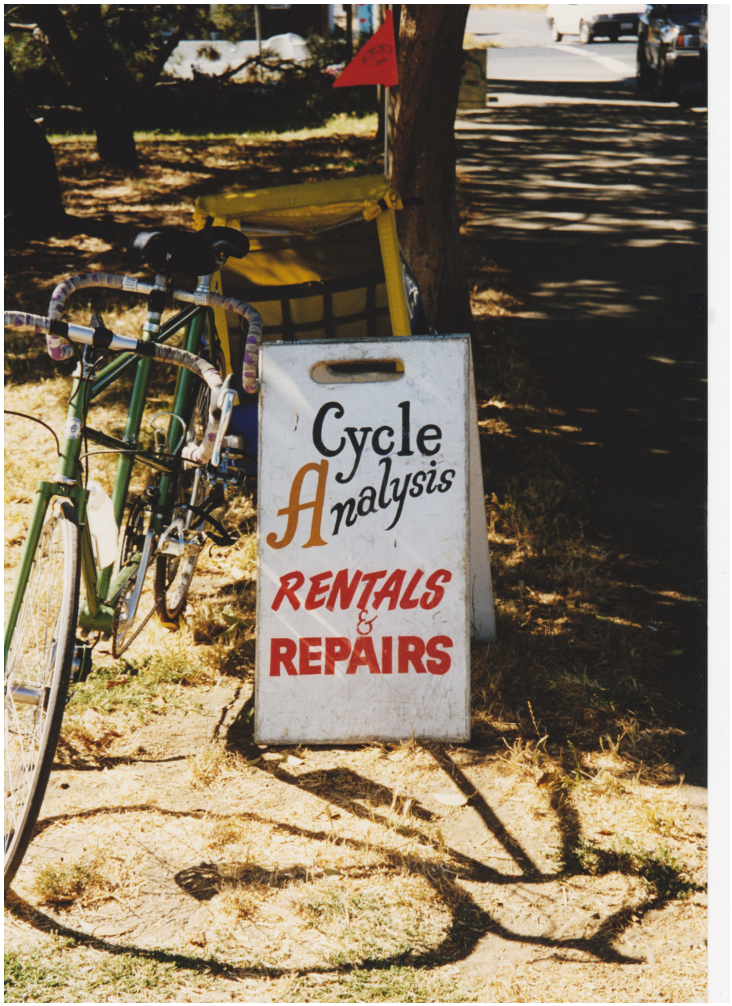
Parce que ce genre de bon cycle se produit naturellement

- Dans la composition de modules
  - cf. séminaire d'Emmanuel Ledinot, cours du 16/04/2013

    http://www.college-de-france.fr/site/gerard-berry/seminar-2013-04-16-11h00.htm
- Dans la conception de circuits efficaces et électriquement corrects
  - gain exponentiel dans le meilleurs cas (rare)
  - meilleure expression des cas symétriques
  - cf. cours du 26/03/2014

    http://www.college-de-france.fr/site/gerard-berry/course-2014-03-26-16h00.htm

… et parce que la solution est connue !



voir cours
du 26/03/2014

Mendler-Shiple-Berry : équivalence entre cycles électriquement sains et logique intuitionniste (constructive)

Il faut aimer et protéger les bons cycles de causalité

D'après Cavanna

Agenda

1. Quelle type de sémantique choisir ?
2. Rappel : le noyau d'Esterel Pur
3. Codes de retour : terminaison, exits
4. La sémantique comportementale (Berry − Cosserat, 1984)
5. Problèmes de causalité
6. Il faut aimer et protéger les bons cycles de causalité !
7. Vers la sémantique constructive

La sémantique constructive

Au lieu de résoudre des équations logiques,
se contenter de propager des faits prouvés
(principe de la logique intuitionniste)

"If can, can. If no can, no can." — Hawaiian pidgin proverb

merci à Robby Findler et Spencer Florence

Propagation constructive : Must et Cannot

```
|| emit X
|| if X else emit Y
|| if Y then emit Z
```

$X^{\perp}\ Y^{\perp}\ Z^{\perp}$

Must

```
|| emit X
|| if X else emit Y
|| if Y else emit Z
```

$X^{+}\ Y^{\perp}\ Z^{\perp}$

Cannot

```
|| emit X
|| if X else emit Y
|| if Y else emit Z
```

$X^{+}\ Y^{-}\ Z^{\perp}$

car il n'y a plus d'émetteurs de Y

Must

```
|| emit X
|| if X else emit Y
|| if Y else emit Z
```

$X^{+}\ Y^{-}\ Z^{+}$

Les règle de signal mise à jour

"If must, do. If no can, no do." —Esterel pidgin proverb

$$\frac{S \in Must_s(p, E \cup \{S^\perp\}) \qquad p \xrightarrow[E \cup \{S^+\}]{E', k} p'}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[E]{E' \setminus \{S^+\}, k} \delta^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})} \quad (\text{csignal}+)$$

$$\frac{S \notin Can_s^+(p, E \cup \{S^\perp\}) \qquad p \xrightarrow[E \cup \{S^-\}]{E', k} p'}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[E]{E' \setminus \{S^-\}, k} \delta^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})} \quad (\text{csignal}-)$$

Bien plus simple que (signal++) et (signal--)
… à condition de savoir calculer *Must* et *Can*

Les fonctions Must et Can^m

- Signaux augmentés:

  S^+ : S connu comme émis

  S^- : S connu comme non émis

  $S^\perp$: S inconnu

  E : ensemble de signaux augmentés

- *Must*(p, E) = (E', K)

  ne s'applique qu'aux les termes devant être exécutés

  E' : signaux devant être émis

  K : vide ou réduit au code de retour garanti rendu

- *Can*^{m}(p, E) = (E', K)

  s'applique à tous les termes

  m = + si l'on sait que p doit être exécuté, m = $\perp$ sinon

  E' : signaux pouvant être émis,

  K : codes de retour pouvant êtres rendus

Itération par déduction positive

```
p : signal Z in
q :    emit Z ; if Z then emit X else emit Y
   end
```

Si l'on sait déjà qu'on doit exécuter p dans $\{X^{\perp}, Y^{\perp}\}$:

1. premier calcul : $Must(q, \{X^{\perp}, Y^{\perp}, Z^{\perp}\}) = (\{Z^{+}\}, \emptyset)$

   $Can^{+}(q, \{X^{\perp}, Y^{\perp}, Z^{\perp}\}) = (\{X^{+}, Y^{+}, Z^{+}\}, \{0\})$

Puisque Z doit être émis, on peut itérer :

2. recalcul : $Must(q, \{X^{\perp}, Y^{\perp}, Z^{+}\}) = (\{X^{+}, Z^{+}\}, \{0\})$

   $Can^{+}(q, \{X^{\perp}, Y^{\perp}, Z^{+}\}) = (\{X^{+}, Z^{+}\}, \{0\})$

Donc X doit être émis, Y ne peut pas être émis,
q et p doivent terminer

Itération par déduction négative

```
p : signal Z in
q :    if Z then emit X else emit Y
   end
```

Si l'on sait déjà qu'on doit exécuter p dans $\{X^{\perp}, Y^{\perp}\}$:

1. premier calcul : $Must(q, \{X^{\perp}, Y^{\perp}, Z^{\perp}\}) = (\emptyset, \emptyset)$
   $Can^{+}(q, \{X^{\perp}, Y^{\perp}, Z^{\perp}\}) = (\{X^{+}, Y^{+}\}, \{0\})$

Puisque Z ne peut pas être émis, on peut itérer

2. recalcul : $Must(q, \{X^{\perp}, Y^{\perp}, Z^{-}\}) = (\{Y^{+}\}, \{0\})$
   $Can^{+}(q, \{X^{\perp}, Y^{\perp}, Z^{-}\}) = (\{Y^{+}\}, \{0\})$

Donc Z ne peut pas être émis, X non plus ici,
Y doit être émis, et q et p doivent terminer

Incertitude provisoire

```
p : signal Z in
q :    emit Z ; if Z then emit X else emit Y
   end
```

Si l'on ne sait pas encore si doit exécuter p

1. premier calcul : $Must(p,\{X^{\perp},Y^{\perp},Z^{\perp}\}) = (\emptyset,\emptyset)$

$Can^{\perp}(p,\{X^{\perp},Y^{\perp},Z^{\perp}\}) = (\{X^{+},Y^{+},Z^{+}\},\emptyset)$

On ne peut rien déduire de plus pour l'instant

$Can^{\perp}$: Halte à la spéculation !

```
p : signal Z in
q :    emit Z ; if Z then emit X else emit Y
   end
```

Si l'on ne sait pas encore si doit exécuter *p* ou pas

1. premier calcul : *Must* pas encore appelé

$$Can^{\perp}(p, \{X^{\perp}, Y^{\perp}, Z^{\perp}\}) = (\{X^{+}, Y^{+}\}, \emptyset)$$

On ne doit pas spéculer sur l'émission de Z,
et donc rien déduire de plus !

$Can^{\perp}$ indique qu'il ne faut pas itérer
Must et *Can* comme pour Can^{+},
sous peine de trouver des faux négatifs.

La bonne nouvelle

1. Tous les programmes bien causaux sont acceptés
2. Tous les programmes pas bien causaux sont rejetés

Rejet des programmes non causaux

p : if S then emit S end

q : if S else emit S end

r : if S1 else emit S2 end || if S2 then emit S1 end

h : if S then emit S else emit S end

Must (p,E) = *Must* (q,E) = *Must* (r,E) = *Must* (h,E) = (∅,∅)

Can^m (p,E) = Can^m (q,E) = Can^m (h,E) = ({S},∅)

Can^m (r,E) = ({S1, S2},∅)

Les quatre cas sont traités de la même manière !

Agenda

1. Quelle type de sémantique choisir ?
2. Rappel : le noyau d'Esterel Pur
3. Codes de retour : terminaison, exits
4. La sémantique comportementale (Berry − Cosserat, 1984)
5. Problèmes de causalité
6. Il faut aimer et protéger les bons cycles de causalité !
7. Vers la sémantique constructive
8. Le calcul de Can et Must

Cas de base

$$
\begin{aligned}
\mathit{Must}\,(\texttt{nothing}, E) &= \mathit{Can}^m(\texttt{nothing}, E) &&= \langle \emptyset, \{0\} \rangle \\
\mathit{Must}\,(\texttt{pause}, E) &= \mathit{Can}^m(\texttt{pause}, E) &&= \langle \emptyset, \{1\} \rangle \\
\mathit{Must}\,(\texttt{exit}\ T^k, E) &= \mathit{Can}^m(\texttt{exit}\ k, E) &&= \langle \emptyset, \{k\} \rangle \\
\mathit{Must}\,(\texttt{emit}\ S, E) &= \mathit{Can}^m(\texttt{emit}\ S, E) &&= \langle \{S^+\}, \{0\} \rangle
\end{aligned}
$$

Séquence constructive

$$Must(p;q,E) = \begin{cases} Must(p,E) \text{ si } 0 \notin Must_k(p,E) \\ \langle Must_s(p,E) \cup Must_s(q,E), Must_k(q,E) \rangle \\ \quad \text{si } 0 \in Must_k(p,E) \end{cases}$$

$$Can^m(p;q,E) = \begin{cases} Can^m(p,E) \text{ si } 0 \notin Can^m_k(p,E) \\ \langle Can^m_s(p,E) \cup Can^{m'}_s(q,E), Can^m_k(p,E) \cup Can^{m'}_k(q,E) \rangle \\ \quad \text{si } 0 \in Can^m_k(p,E) \\ \quad \text{avec } m' = + \text{ si } m = + \text{ et } 0 \in Must_k(p,E), \\ \qquad m' = \bot \text{ sinon} \end{cases}$$

Halte à la spéculation !

Dans E, q ne doit pas encore être exécuté, mais peut-être le sera-t-il ultérieurement dans un environnement E' où plus de signaux seront définis

Test constructif

$$Must(\texttt{if } S \texttt{ then } p \texttt{ else } q \texttt{ end}, E) = \begin{cases} Must(p, E) \text{ si } S^+ \in E \\ Must(q, E) \text{ si } S^- \in E \\ \langle \emptyset, \emptyset \rangle \text{ si } S^\perp \in E \end{cases}$$

$$Can^m(\texttt{if } S \texttt{ then } p \texttt{ else } q \texttt{ end}, E) = \begin{cases} Can^m(p, E) \text{ si } S^+ \in E \\ Can^m(q, E) \text{ si } S^- \in E \\ Can^\perp(p, E) \cup Can^\perp(q, E) \\ \quad \text{si } S^\perp \in E \end{cases}$$

Parallèle constructif

$$Must\,(p \mid\mid q, E) = \langle\, Must_s\,(p, E) \cup Must_s\,(q, E)\,, \; setmax(Must_k\,(p, E), Must_k\,(q, E))\,\rangle$$

$$Can^m(p \mid\mid q, E) = \langle\, Can^m_s(p, E) \cup Can^m_s(q, E)\,, \; setmax(Must_k\,(p, E), Must_k\,(q, E))\,\rangle$$

avec $setmax(K, K') = \{max(k, k') \mid k \in K, k' \in K'\}$

Suspend, loop et trap constructifs

$$Must\,(\texttt{suspend}\ S\ \texttt{when}\ p, E) = Must\,(p, E)$$
$$Can^m(\texttt{suspend}\ S\ \texttt{when}\ p, E) = Can^m(p, E)$$

$$Must\,(\texttt{loop}\ p\ \texttt{end}, E) = Must\,(S, p)$$
$$Can^m(\texttt{loop}\ p\ \texttt{end}, E) = Can^m(p, E)$$

$$Must\,(\texttt{trap}\ T^k\ \texttt{in}\ p\ \texttt{end}, E) = \langle Must_s\,(p, E), \downarrow Must_k\,(p, E)\rangle$$
$$Can^m(\texttt{trap}\ T^k\ \texttt{in}\ p\ \texttt{end}, E) = \langle Can^m_s(p, E), \downarrow Must_k\,(p, E)\rangle$$
$$\text{avec}\quad \downarrow K = \{\downarrow k \mid k \in K\}$$

Déclaration de signal constructive

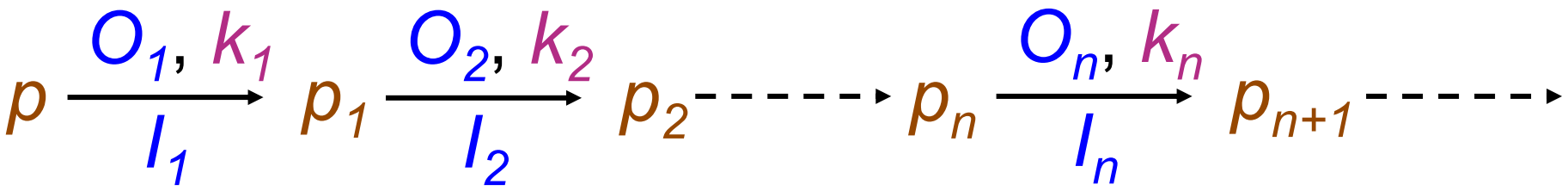
$$Must(\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end}, E) = \begin{cases} Must(p, E \cup \{S^+\}) \\ \quad \text{si } S \in Must_s(p, E \cup \{S^\perp\}) \\ Must(p, E \cup \{S^-\}) \\ \quad \text{si } S \notin Can_s^+(p, E \cup \{S^\perp\}) \\ Must(p, E \cup \{S^\perp\}) \\ \quad \text{sinon} \end{cases}$$

Halte à la spéculation !

$$Can^m(\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end}, E) = \begin{cases} Can^m(p, E \cup \{S^+\}) \\ \quad \text{si } m = + \text{ et } S \in Must_s(p, E \cup \{S^\perp\}) \\ Can^m(p, E \cup \{S^-\}) \\ \quad \text{si } S \notin Can_s^m(p, E \cup \{S^\perp\}) \\ Can^m(p, E \cup \{S^\perp\}) \\ \quad \text{sinon} \end{cases}$$

Sémantique → Interpréteur

Soit I_1, I_2, … , I_n une suite d'entrées. Alors le calcul des réactions successives d'un programme p est facile :



Notes :

1. Les k_i valent 0 ou 1 car p n'a pas de trappes libres.
2. Si $k_m = 0$, l'exécution est terminée et tous les O_n sont vides pour $n>m$

Conclusion

- La sémantique constructive est la référence pour Esterel
- Elle est calculable (écrire les règles en Prolog, Caml, etc.)
- On peut donc l'utiliser pour construire un interpréteur
- Nous verrons que le nombre de p' est fini. On peut donc explorer exhautivement les état et compiler en automates
  - Esterel v2 (1985), Berry-Couronné (1985), tri topologique
  - Esterel v3 (1988), Gonthier → CMA → CISI / ILOG, bien meilleur potentiels = analyse statique approximée de causalité

Mais les automates peuvent être exponentiels
La prochaine fois, allons vers la nanoseconde !
Traduction quasi-linéaire en circuits booléens

Sémantiques opérationnelles

- Cinq approches (*Compiling Esterel* Potop−Edwards−Berry) :
  - Implémentation brutale de la sémantique constructive, inefficace
  - Amélioration opérationnelle dans le cadre SOS (D. Potop)
  - Traduction en circuits booléens, puis implémentation matérielle sur circuits ou logicielle par simulation (cours 4 du 07/03/2018)
  - Implémentation directe en C efficace (S. Edwards, 1999)
- Nouveau :
  - Propagation SOS subtile de jetons colorés dans les termes
    L. Rieg, 2016 (aussi G. Gonthier 1986, négligé par ma faute ?)
  - Mise en Coq du noyau et de ses sémantiques + preuves,
    L. Rieg, séminaire du 28/03/2016
- A venir : preuves en Coq de l'équivalence Jetons ⟺ circuits

Au 7 mars pour les circuits
et au 14 mars pour la réincarnation !