L’APPROCHE REACTIVE-SYNCHRONE

L’approche synchrone

Le langage Esterel

Sémantique formelle...

Systèmes réactifs

Transformationel :

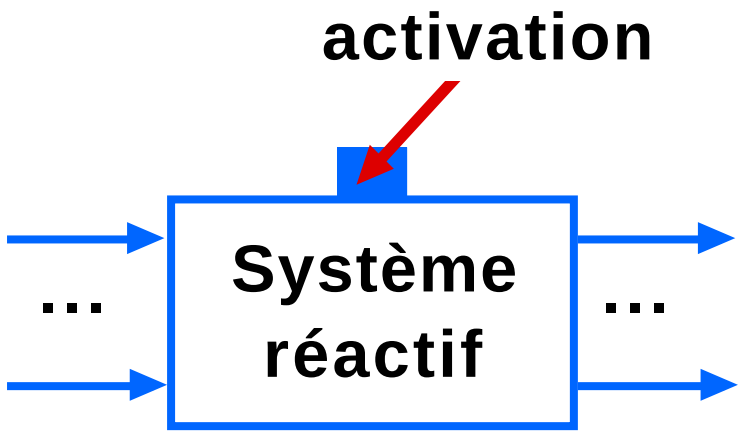
lancement avec paramètres;
exécution;
retour des résultats

Interactif :

```
while(true){
    activation;
    input des paramètres;
    exécution;
    output des résultats
}
```

Réactif :

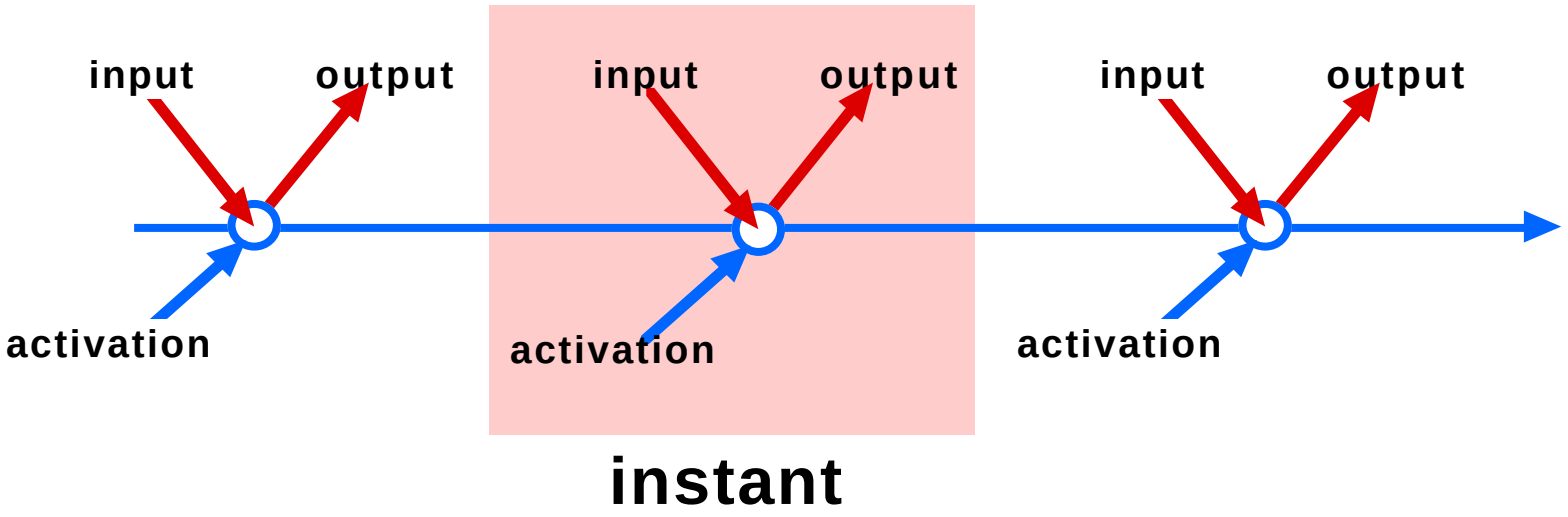
intéractif + toujours prêt à traiter de nouveaux inputs



Systèmes embarqués
IHM
Protocoles
Circuits
...

Approche synchrone

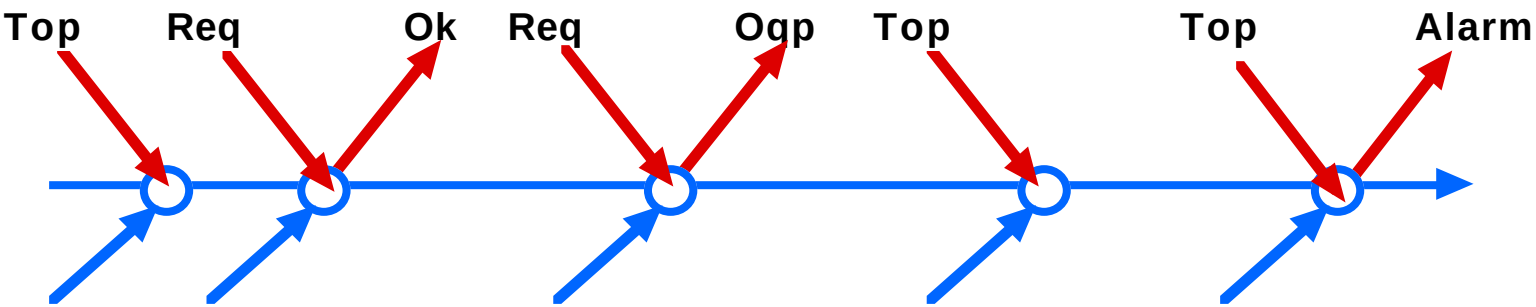
- **Réaction de durée nulle**
  hypothèse idéale pour assurer la réactivité
- **Déterminisme**
  la même suite d'inputs produit la même suite d'outputs
- **Aucune création dynamique**
  pour permettre les preuves et validations formelles



Un exemple

Système d'acquittement Top/Req

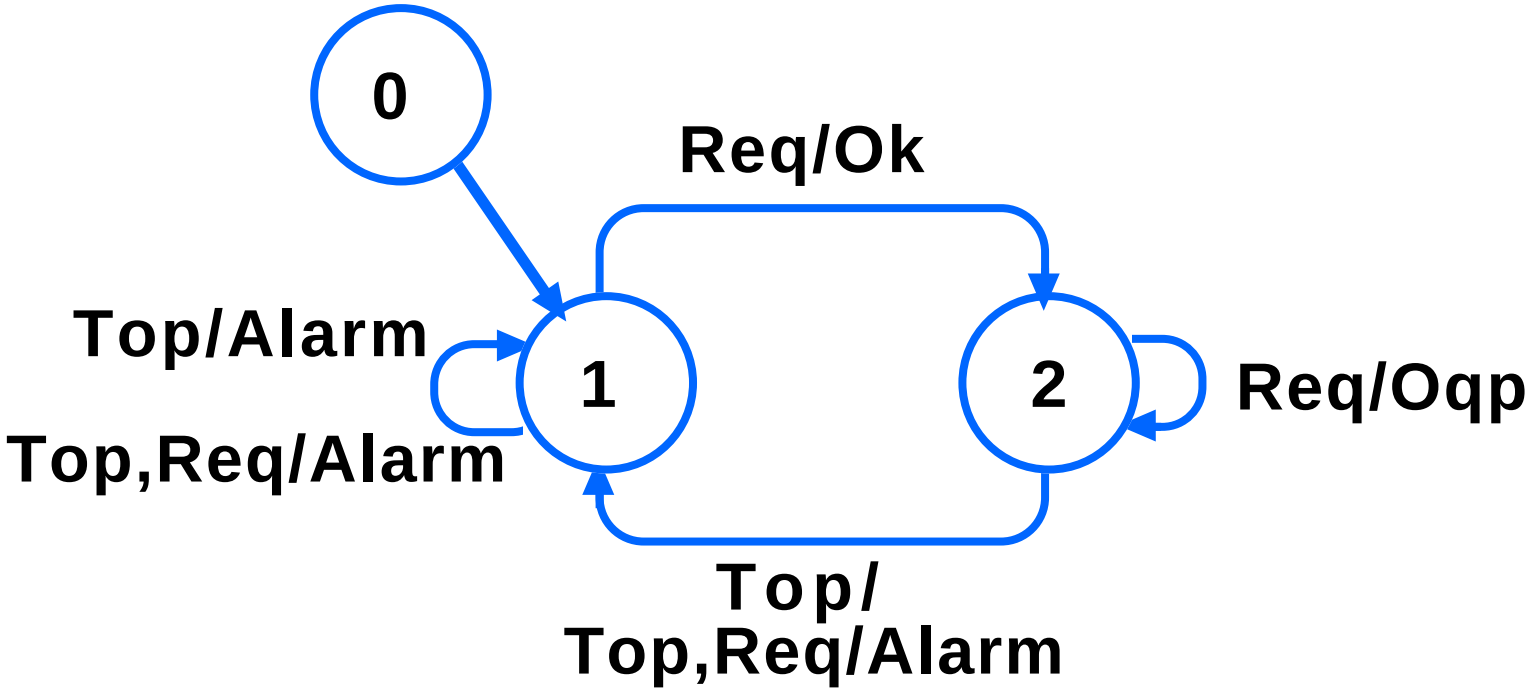
- Premier Req entre deux Tops -> Ok ; Req suivants -> Oqp
- Alarm si aucun Req entre deux Tops



Ambiguité : Top et Req simultanés ?

Automates

Technique standard : les machines d'états finis



Technique monolithique, sans parallélisme.
Automates difficilement réutilisables (ex : reset)

Esterel

Langage de programmation synchrone

- **Style impératif**
- **Opérateur de parallélisme (logique)**
- **Communication par événements diffusés**
- **Centré sur le contrôle**
- **Approche compilée**
- **Environnement de programmation industrialisé par Esterel Technologies**

`http://www.esterel.org`

Instructions réactives

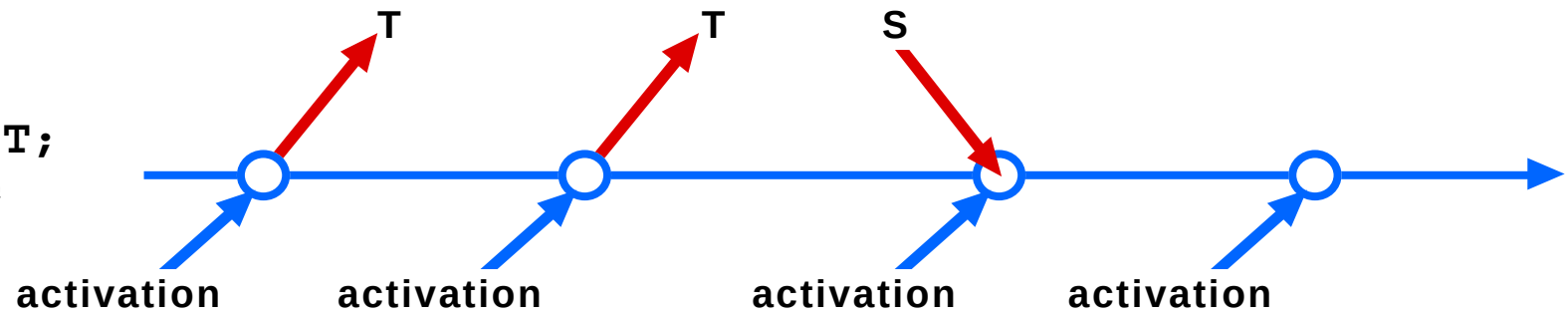
Séquence : “`;`”
Parallélisme : “`||`”
Instant : `pause`
Boucle infinie : `loop ... end`
Boucle finie : `repeat ... times ... end`

Signal : `signal S in ... end`
Emission : `emit S`
Attente : `await S`
Présence : `present S then ... else ... end`

Préemption

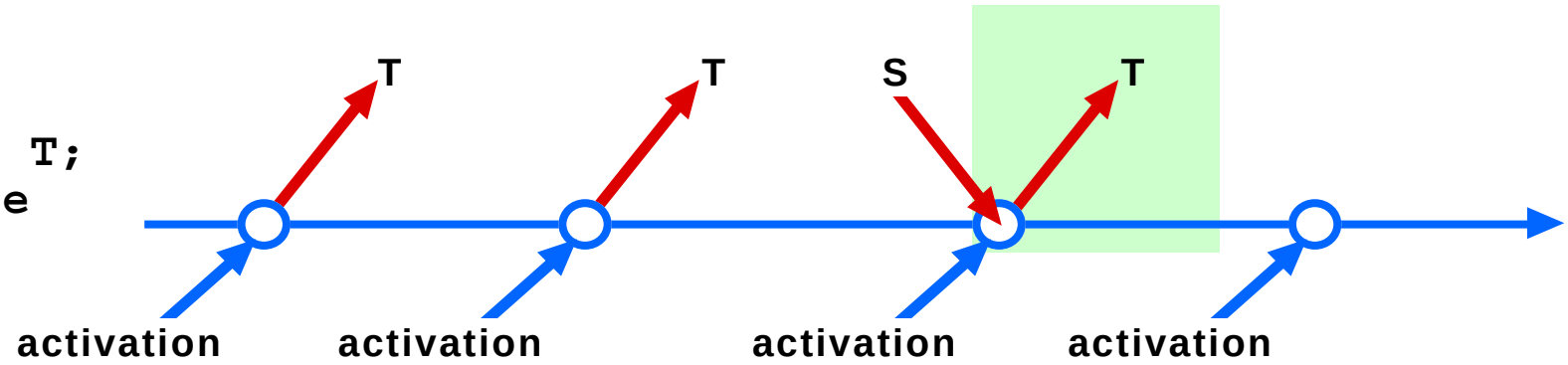
forte : `abort ... when S do ... end`

```
abort
   loop
      emit T;
      pause
   end
when S
```



faible : `weak abort ... when S do ... end`

```
weak abort
   loop
      emit T;
      pause
   end
when S
```



Instructions dérivées

```
halt                    →   loop pause end

sustain S               →   loop
                                emit S;pause
                            end

do ... upto S           →   abort
                                ...; halt
                            when S

every S do ... end      →   loop
                                await S;
                                abort ... when S
                            end
```

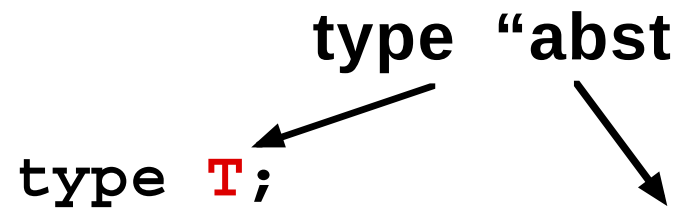
Top/Req

```
module TopReq :
input Top,Req
output Ok,Oqp,Alarm

signal Question,Received in
   every Top do
     emit Question;
     present Received else emit Alarm end
   end
||
   loop
      await Req; emit Ok;
      abort
         every Req do emit Oqp end
      when Question do emit Received end
   end
end

end module
```

Liens avec les données




```
type T;
var x : int, y : T in ... end
x := 3; y := fun(...)
if ... then ... else ... end

call proc(r1,r2)(v1,v2)
```

Signaux valués

```
signal S : integer with plus
emit S(3)
emit S(3);emit S(4)
x := ?S
```

Correspondance avec les threads ?

Séquence :

```
a;b   join(a);b
```

Très approximatif ...

Parallélisme :

```
a||b   new Thread(a).start();
       new Thread(b).start();
```

Boucle : `loop a end` `while(true){a}`

Signaux : `emit S, await S` `synchronized(S)`

Préemption : `abort a when S` `a.stop()`

Instants : `pause` `yield()`

Programmation synchrone dataflow

Flot : suite de valeurs + horloge de présence

```
X = 0 1 2 3 4 ...
Y = - 0 1 2 3 ...
Z = 0 - 2 - 4 - ...
```

Syntaxe pour exprimer les flots : Lustre, Signal

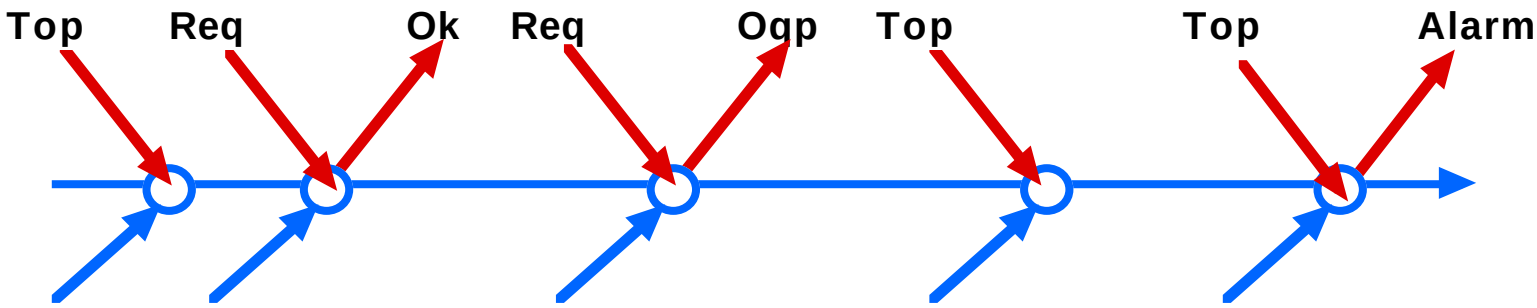
```
X = 0 -> 1 + pre(X)
Y = pre(X)
Altern = true -> not(pre(Altern))
Z = X when Altern
```

Calculs d'horloges : assure une solution unique au système d'équations + mémoire bornée

```
T = X + Z
U = U + 1
V = 0 -> V
W = pre(W)
```

TopReq en dataflow

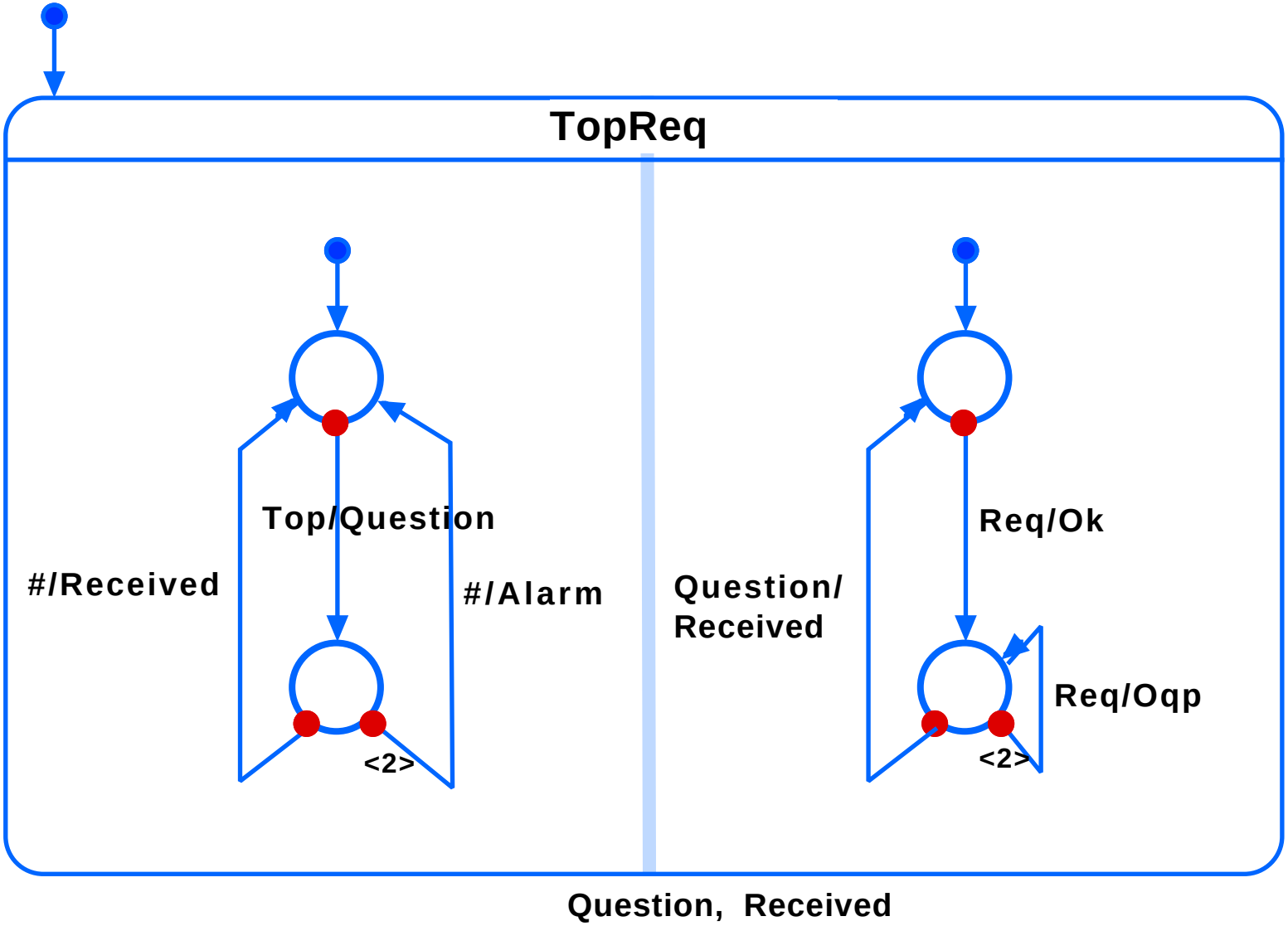
```
node TopReq (Req, Top : bool)
returns (Alarm : bool);
var Ok,Oqp : bool;
let
   Ok = false -> Req and not pre(Req);
   Oqp = false -> Req and pre(Req);
   Alarm = false ->
           Top and not pre(Ok) and not pre(Oqp);
tel.
```



Top	v	f	f	v	v
Req	f	v	v	f	f
Ok	f	v	f	f	f
Alarm	f	f	f	f	v

Programmation synchrone graphique

Statecharts, Argos, SyncCharts



Problèmes...

En Esterel :

```
loop x := 1 end
```

pas de solution

```
emit S(?S + 1)
```

```
signal S in
   present S else emit S end
end
```

“problème de causalité” :
l’émission de S est causée par son absence

Problèmes - 2

```
emit S(?S)
```

plusieurs solutions

```
signal S in
   present S then emit S end
end
```

```
signal S1,S2 in
   present S1 then emit S2 end
||
   present S2 then emit S1 end
end
```

problème pour la modularité

```
run S1thenS2 || run S2thenS1
```

Causalité et préemption

```
abort
   pause;
   emit S
when S
```

```
abort
   pause;
   emit T
when S
```

||

```
abort
   loop pause end
when T do
   emit S
end
```

Pas de problème avec la préemption faible

Problème = réaction instantanée à l'absence

Solution unique

```
signal S in
   present S then emit T end
||
   emit S
end
```

L'unicité de la solution pas suffisante, car il y a des solutions non causales

```
signal S in
   present S then emit T end;
   emit S
end
```

Problème d'efficacité : comment éviter d'essayer toutes les solutions ? (2^N)

Les approches de compilation

- Analyse des cycles de dépendances dans le graphe du programme (v4)
- Potentiels (v3)
- Causalité constructive (v5)

rejeté par v4

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

rejeté par v3

```
emit S;
present T then
   present S else emit T end
end
```

rejeté par v5

```
present S then
   present S else emit S end
end
```

Conclusion

Les points forts de l'approche synchrone :

- **grande puissance expressive (mais pas de création dynamique)**
- **possibilité de preuves et validation**
- **production de circuits**

Les faiblesses :

- **problèmes de causalité obstacle à la modularité**
- **pas d'objets**
- **liaison "abstraite" avec les données**

Besoin d'une approche formelle de la sémantique

Références

Systèmes réactifs et programmation synchrone, Charles André, Cours de DEA, 1997.

Synchronous Programming of Reactive Systems, Nicolas Halbwachs, Kluwer Academic Pub., Amsterdam, 1993

The Esterel v5 Primer, Gérard Berry, http://www.inria.fr/meije/esterel