Esterel et SCADE, de la recherche à l'industrie

1. La vision labo

Gérard Berry

Collège de France
Chaire Algorithmes, machines et langages
gerard.berry@college-de-france.fr

Cours 1, Inria Sophia-Méditerranée, 15/01/2014

Hommage à Gilles Kahn et Paul Caspi

Gilles Kahn

Paul Caspi Daniel Pilaud

Agenda

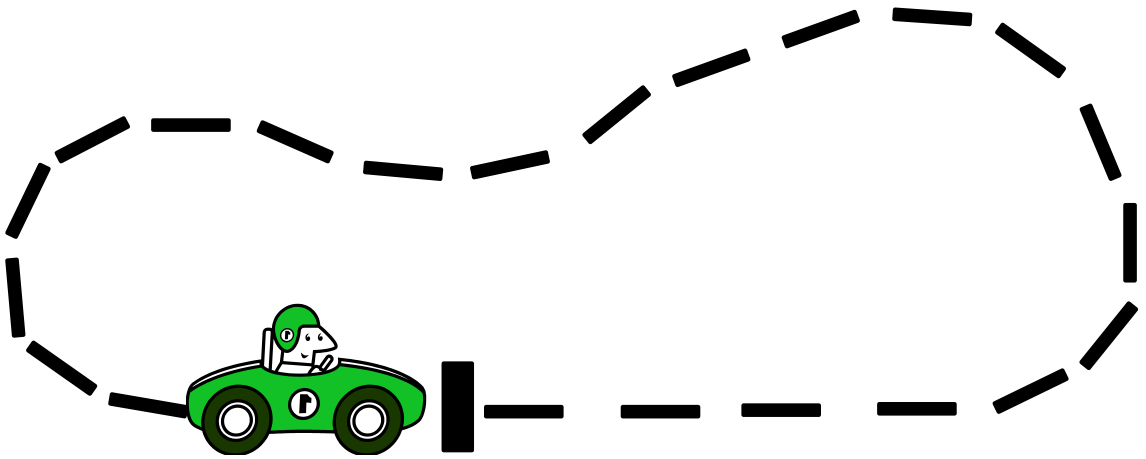
1. Programmer comme on pense
2. Donner un sens aux programmes
3. Implémenter et vérifier
4. Les utilisateurs pionniers
5. Les débuts industriels
6. Le choc électrique

Agenda

1. Programmer comme on pense
2. Donner un sens aux programmes
3. Implémenter et vérifier
4. Les utilisateurs pionniers
5. Les débuts industriels
6. Le choc électrique

Esterel : la naissance

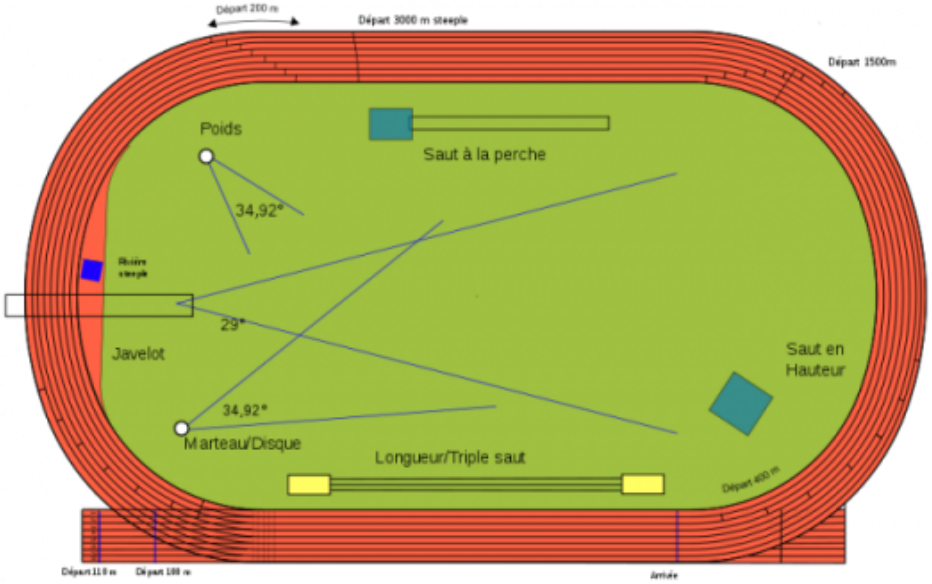
- 1982 : concours Micro Systèmes, naissance de l'idée
  - J-P. Marmorat, J-P. Rigault, J-M. Tanzi
- 1983 : premier design du langage
  - S. Moisan, J. Camerini



1 tour libre

2 chronométrés

Temps multiforme hiérarchique



Second → Hour → Morning

Meter → Lap

Step

HeartBeat → HeartAttack

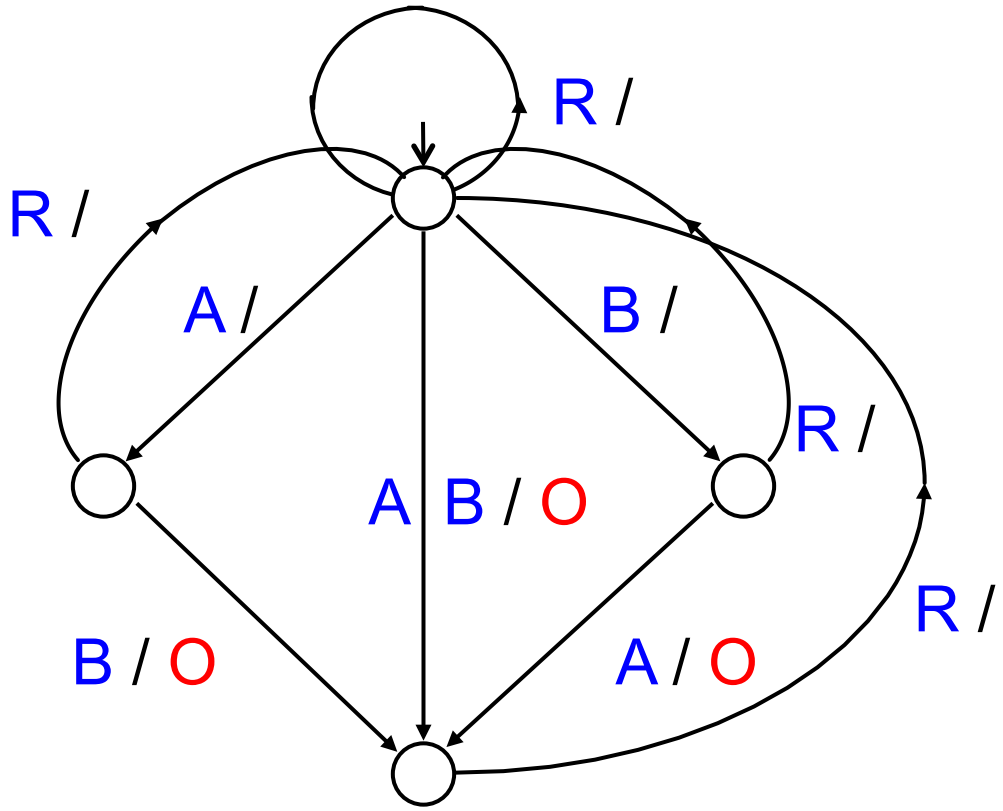
Programmer comme on pense

```
trap HeartAttack in
    every Morning do
        abort
            loop
                abort  run Slowly  when 100 Meter ;
                abort
                    every Step do
                        run Jump || run Breathe || CheckHeart
                    end every
                when 15 Second ;
                run FullSpeed
            each  Lap
        when  4 Lap
    end every
handle HeartAttack fo
   run RushToHospital
end trap
```

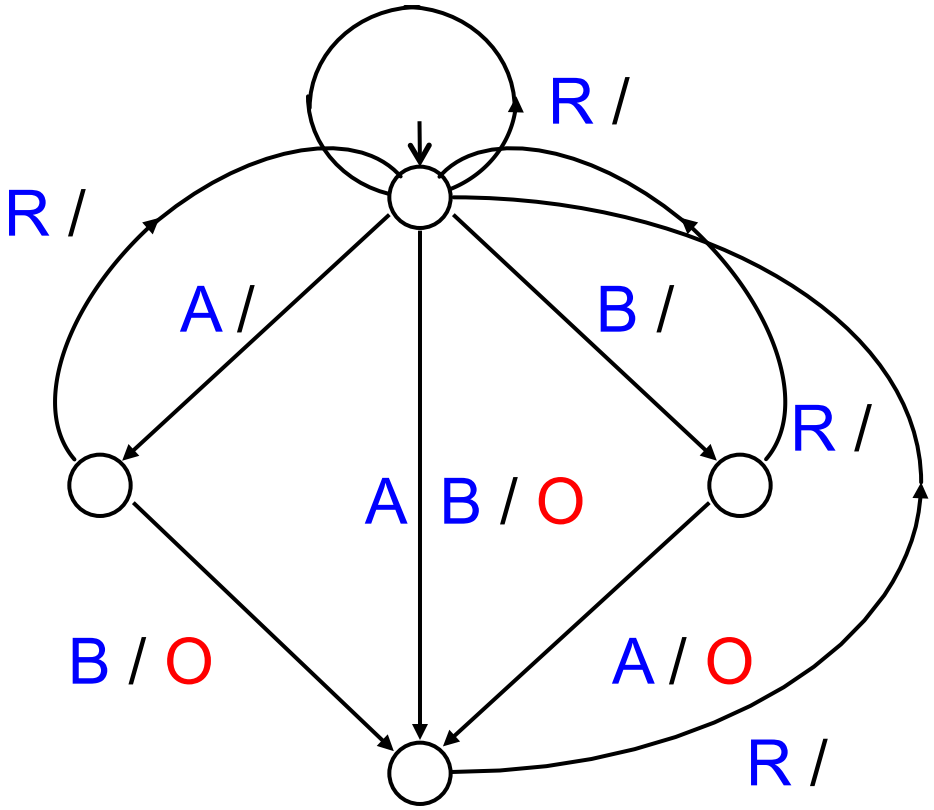
exit HeartAttack

ABRO, le Fibonacci d'Esterel

Emettre O dès que A et B sont arrivés
Réinitialiser le comportement à chaque R

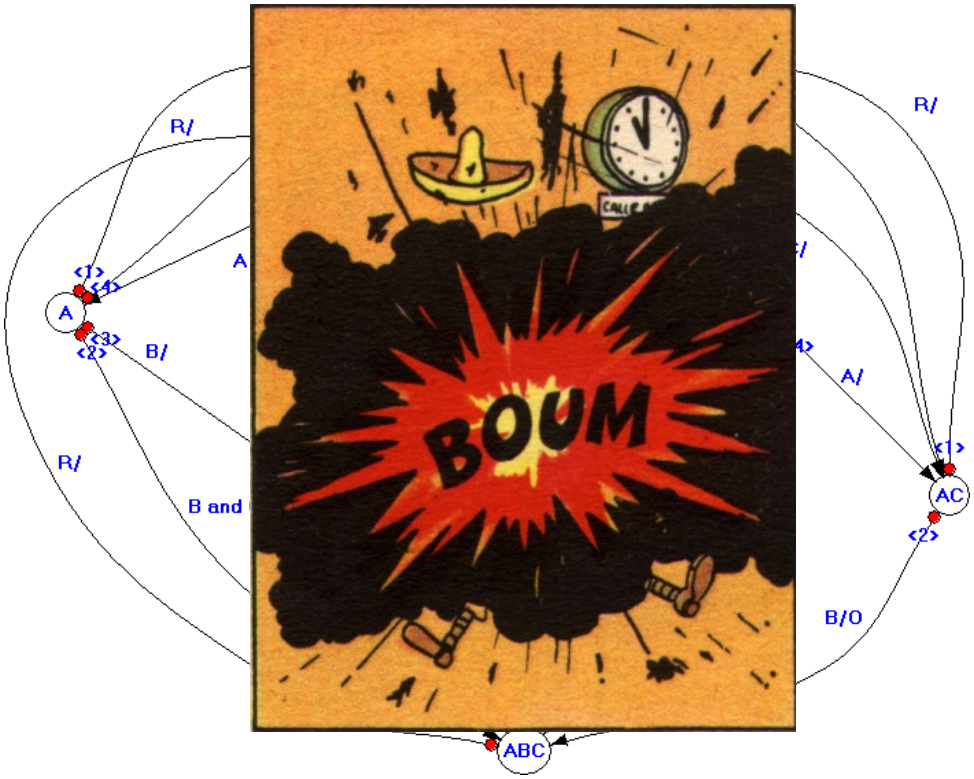


ABRO en Esterel




```
loop
   abort
        { await A || await B };
        emit O;
        halt
   when R
end loop
```

Esterel = spécification linéaire



ABCRO

```
loop
   abort
        {    await A
          || await B
          || await C
        };
        emit O;
        halt
   when R
end loop
```

source: l'oreille cassée, Hergé

Sujet nouveau => problèmes de design

- Faut-il un temps absolu (tick) ou pas ?
- Que veut dire « await S » avec S présent maintenant ?

1. « await S » termine immédiatement

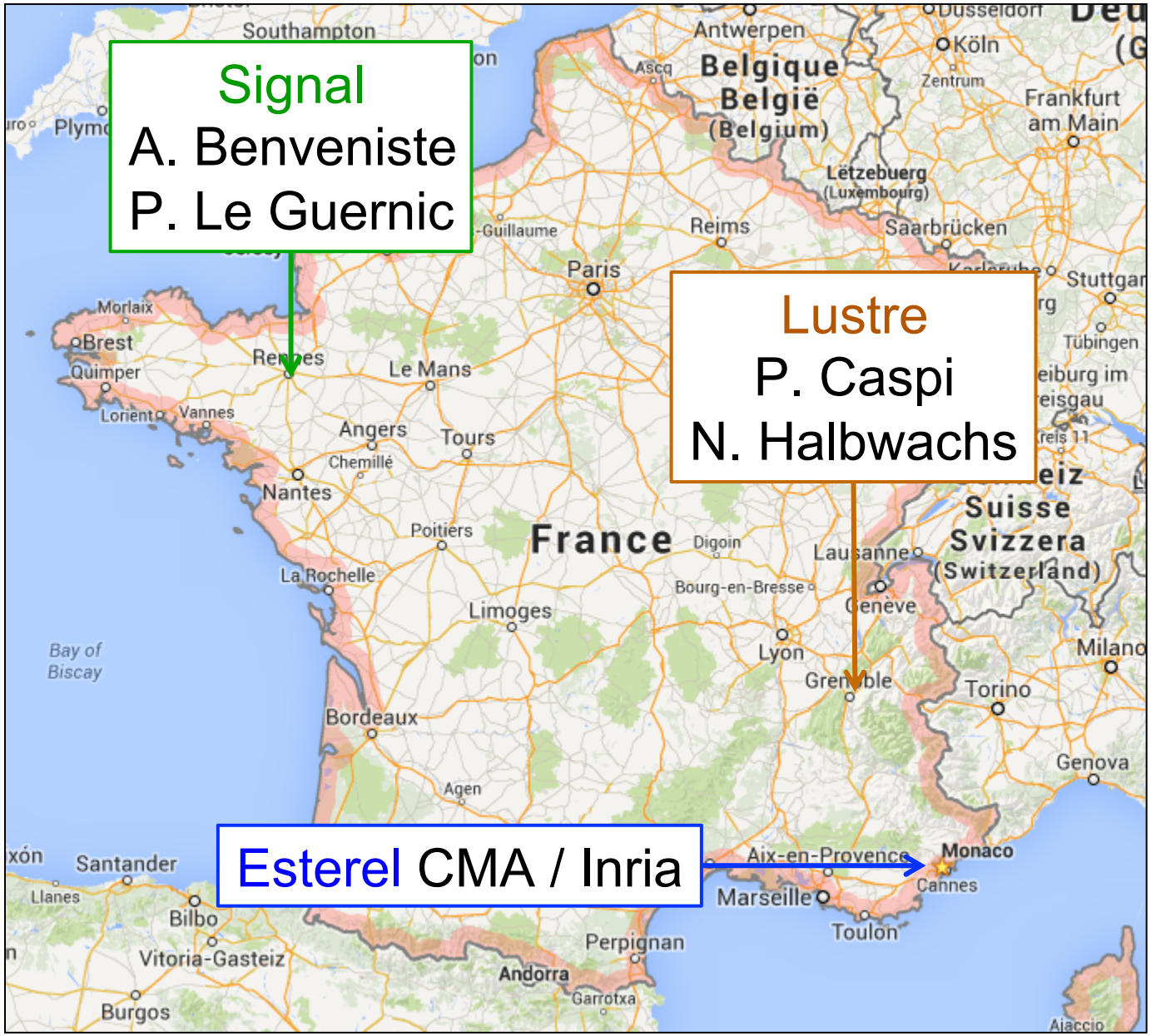
   ⇒ await S; await S ~ await S

   ⇒ await next S; await next S

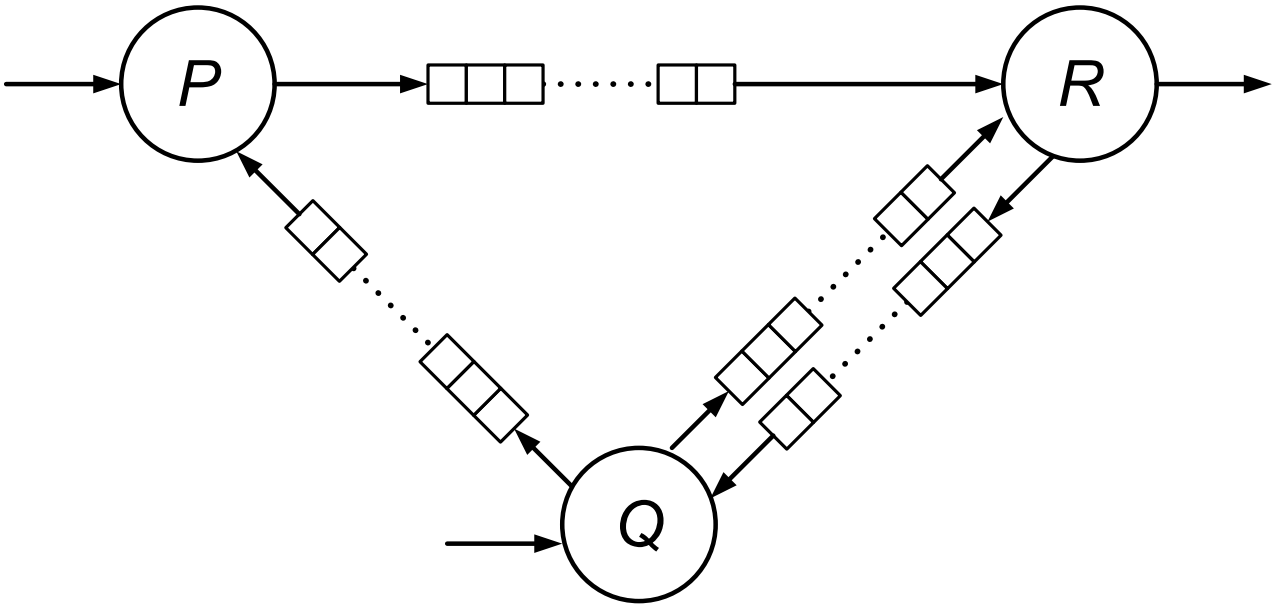
1. « await S » attend le prochain S

   await S ; await S ≈ await 2 S

   await immediate S; await immediate S
   ≈ await immediate S


Signal
A. Benveniste
P. Le Guernic
Lustre
P. Caspi
N. Halbwachs
Esterel CMA / Inria
Southampton
Antwerpen
Belgique
België
(Belgium)
Köln
Frankfurt am Main
Lëtzebuerg
(Luxembourg)
Saarbrücken
Reims
Paris
Stuttgart
Tübingen
Morlaix
Brest
Quimper
Rennes
Le Mans
Lorient
Vannes
Angers
Tours
Chemillé
Nantes
Poitiers
France
Digoin
Lausanne
Suisse
Svizzera
(Switzerland)
Bourg-en-Bresse
Genève
La Rochelle
Limoges
Lyon
Milano
Torino
Grenoble
Bay of Biscay
Bordeaux
Genova
Agen
Santander
Bilbo
Vitoria-Gasteiz
Burgos
Perpignan
Andorra
Garrotxa
Aix-en-Provence
Monaco
Cannes
Marseille
Toulon
Ajaccio

Réseaux de Kahn : asynchronisme



- nœuds déterministes, files non bornées
- ordonnancement non-déterministe arbitraire
- Mais résultat déterministe !

Magnifique sémantique mathématique

Lustre = réseaux de Kahn synchrones

Compteur d'événements

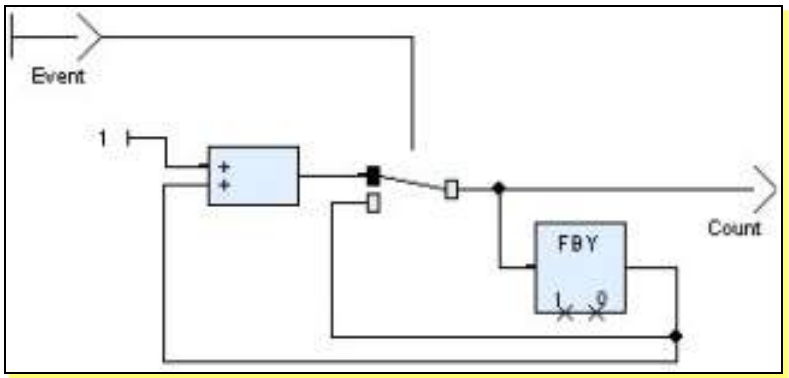
Event = false true false true true false false false true true false

Count = 0 1 1 2 3 3 3 3 4 5 5

$$\begin{cases} Count(0) = 0 \\ \forall t > 0,\ Count(t) = \begin{cases} Count(t-1) + 1, & if\ Event(t) = true \\ Count(t-1), & otherwise \end{cases} \end{cases}$$

```
Count = 0 → (if Event
              then pre(Count)+1
              else pre(Count))
```

Lustre

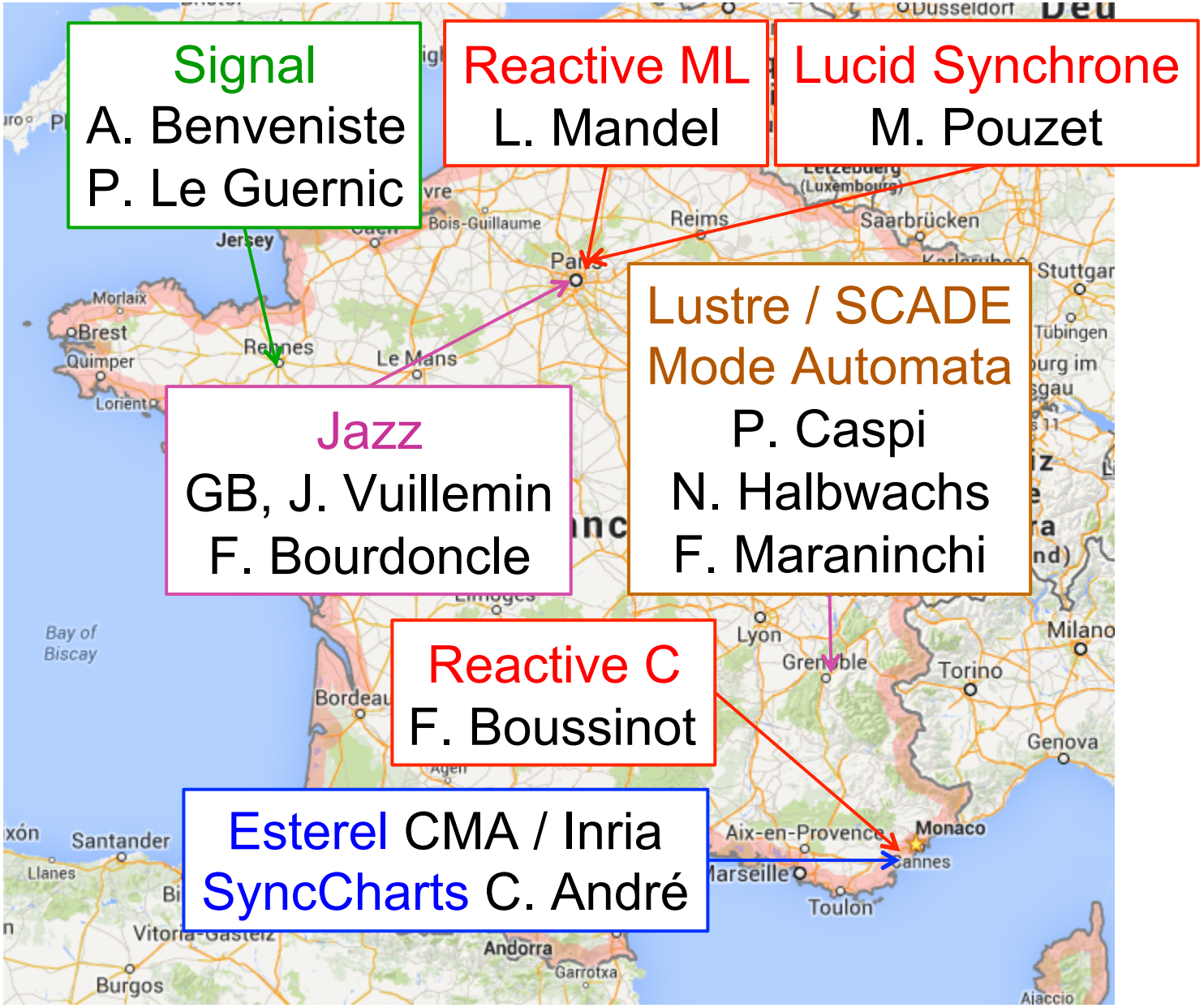


SCADE

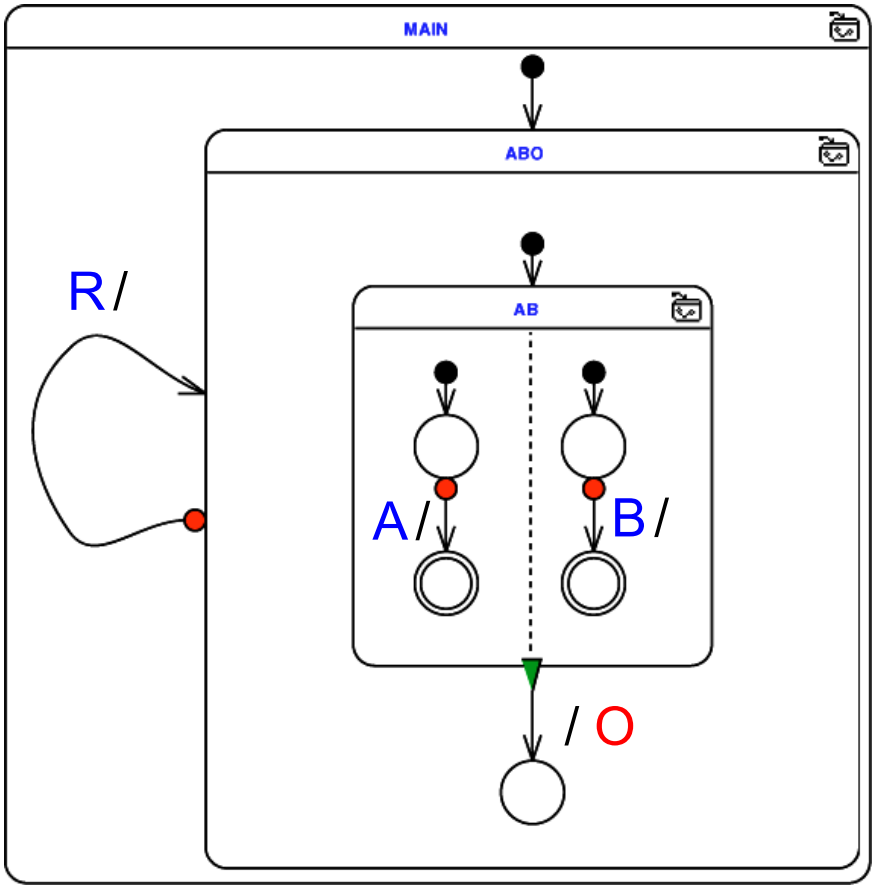
Lustre = langage fonctionnel hiérarchique

```
node Counter (Event, Reset : bool) returns Count : int ;
let
    Count = if (true → Reset) then 0
            else if Event then pre(Count)+1
            else pre(Count))
tel
```

```
Minute = Counter(Second, pre(Minute) = 59) ;
Hour = Counter(Minute, pre(Hour) = 23) ;
```


Signal
A. Benveniste
P. Le Guernic
Reactive ML
L. Mandel
Lucid Synchrone
M. Pouzet
Lustre / SCADE
Mode Automata
P. Caspi
N. Halbwachs
F. Maraninchi
Jazz
GB, J. Vuillemin
F. Bourdoncle
Reactive C
F. Boussinot
Esterel CMA / Inria
SyncCharts C. André
Jersey
Reims
Saarbrücken
Stuttgart
Tübingen
Morlaix
Brest
Quimper
Lorient
Rennes
Le Mans
Bay of Biscay
Bordeaux
Lyon
Grenoble
Torino
Milano
Genova
Monaco
Aix-en-Provence
Marseille
Toulon
Cannes
Santander
Llanes
Vitoria-Gasteiz
Burgos
Andorra
Garrotxa
Aiaccio

ABRO en SyncCharts (C. André)




```
loop
   abort
        { await A || await B };
        emit O;
        halt
   when R
end loop
```

automates parallèles
hiérarchiques synchrones
(Statecharts synchrones)

Linéaires !

Agenda

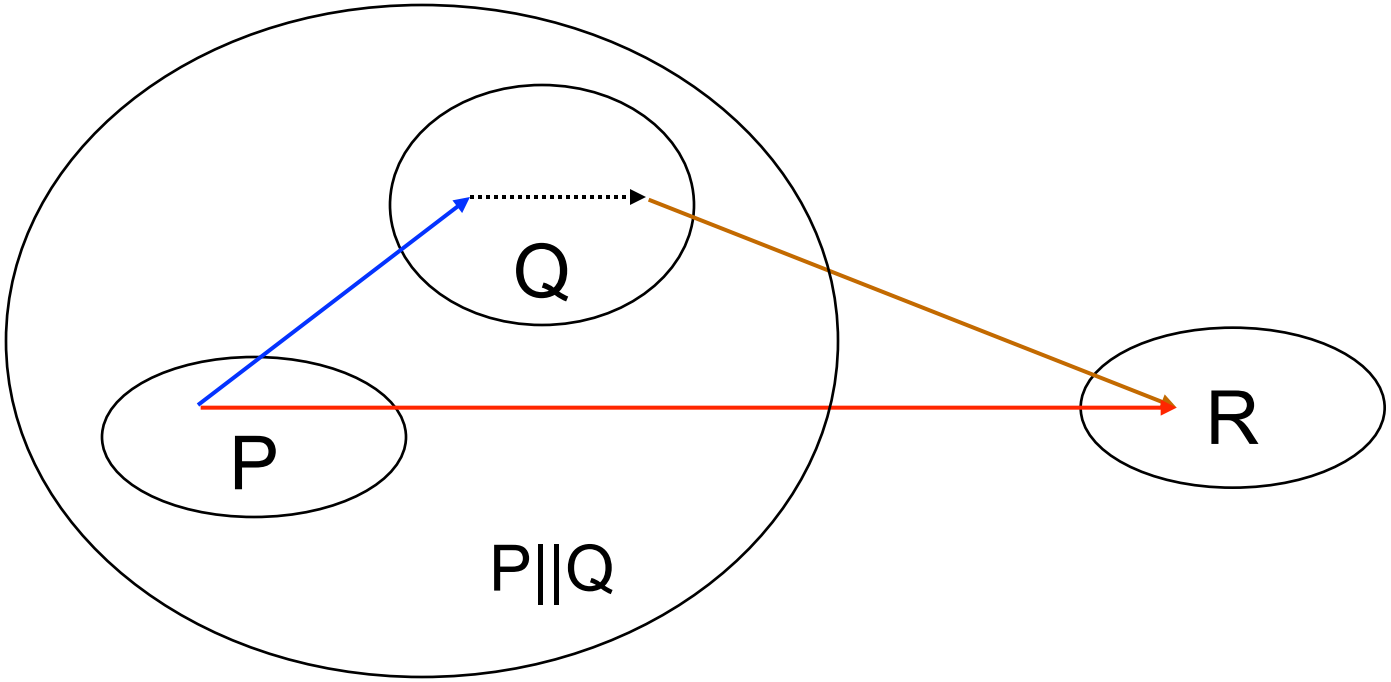
1. Programmer comme on pense
2. Donner un sens aux programmes
3. Implémenter et vérifier
4. Les utilisateurs pionniers
5. Les débuts industriels
6. Le choc électrique

Le principe de synchronisme parfait

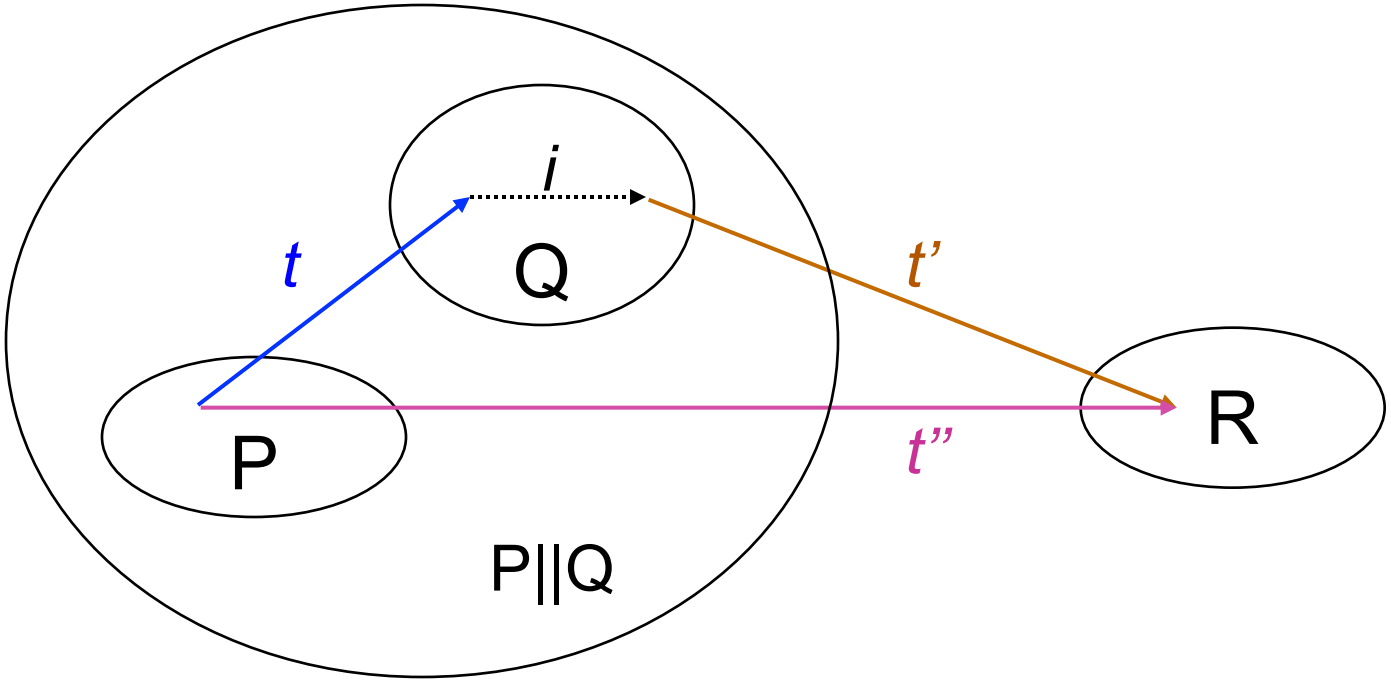
- Comment marchent les instructions temporelles ?
  - séquence « ; », parallélisme « || »
  - chiens de garde « abort-when », « every », etc.
  - communication « emit - await » entre codes parallèles

Idée clef : pour rendre le temps réel facile,
faire comme si la machine était infiniment rapide

Alors seul prend du temps
ce qui est explicitement déclaré en prendre
et la machine n'interfère jamais avec la spécification



Parallélisme : le principe de compositionnalité



$$t'' = t + i + t'$$

$$t'' \approx t \approx i \approx t'$$

$$\Rightarrow t \approx t + t$$

Exactement 3 solutions (en informatique et en physique)

1. *t* arbitraire	asynchronisme
2. *t* nul	synchronisme
3. *t* prévisible	vibration

arbitraire ⇒ non-déterminisme partout

nul ⇒ déterminisme

prévisible ⇒ quasi-déterminisme

Equivalence synchrone ≈ vibratoire

Synchrone

Musiciens et spectateurs négligent la vitesse du son

Vibratoire

Mais les acousticiens règlent sa propagation

Causalité : logique, statique, dynamique ?

~~present X then nothing else emit X end~~ ~~X = not X~~

~~present X then emit X end~~ ~~X = X~~

non-déterminisme ou non-causalité ?

~~present X then emit X else emit X end~~ ~~X = X or not X~~

déterminisme mais causalité ?

X = X or not X

Esterel : approche mathématique

- 1984 : premières sémantiques, prototype Esterel v1
  - L. Cosserat, F. Boussinot, A. Ressouche
  - techniques sémantiques classiques inopérantes
  - mais SOS de Plotkin ! (Structural Operational Semantics)
  - première compréhension de la causalité Esterel

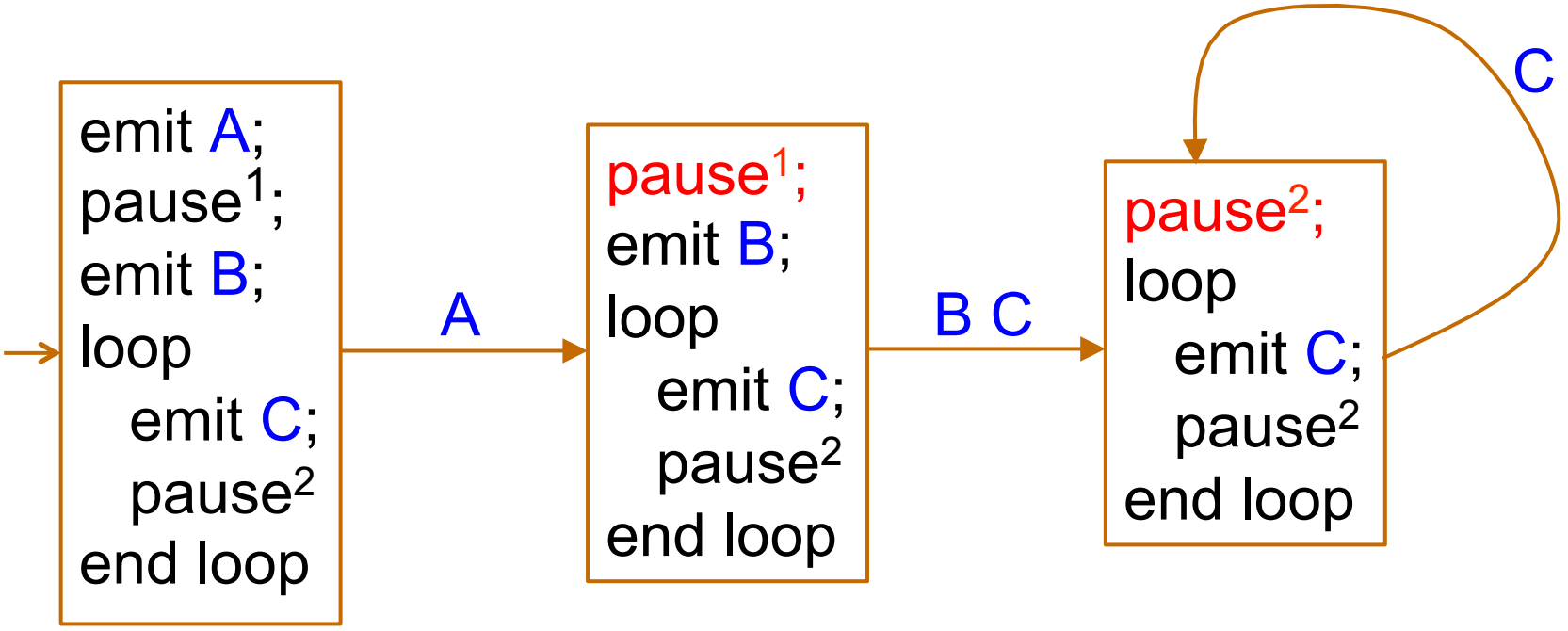
Le sentiment de tenir quelque chose de précieux
mais d'exigeant, demandant de développer
un nouveau corpus théorique et pratique

Agenda

1. Programmer comme on pense
2. Donner un sens aux programmes
3. Implémenter et vérifier
4. Les utilisateurs pionniers
5. Les débuts industriels
6. Le choc électrique

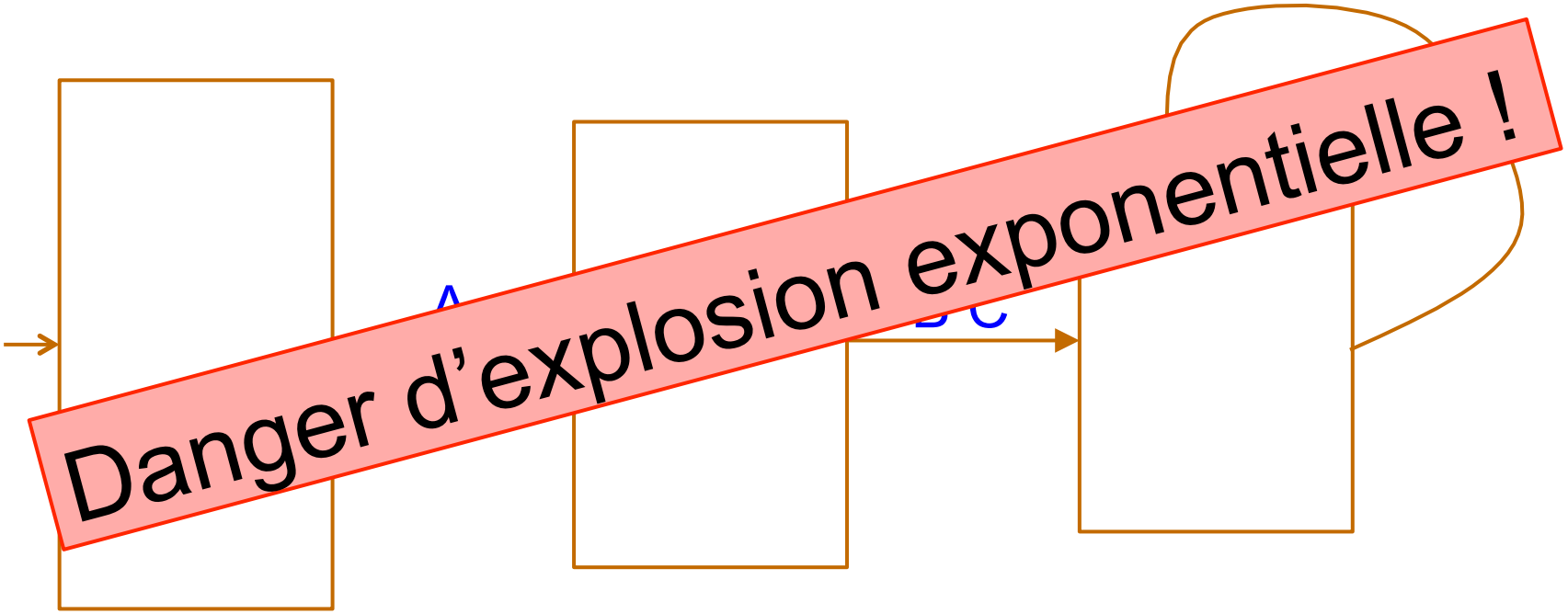
- 1985-6 : Esterel v2, premier vrai compilateur, première vraie distribution
  - résiduelle de Brzozowski : programmes → automates
  - GB, P. Couronné, Le_Lisp sur SM90 (!) puis Sun 3
  - soin particulier pour la bande de distribution et les exemples

Faire directement en vrai, les protos c'est trop cher!

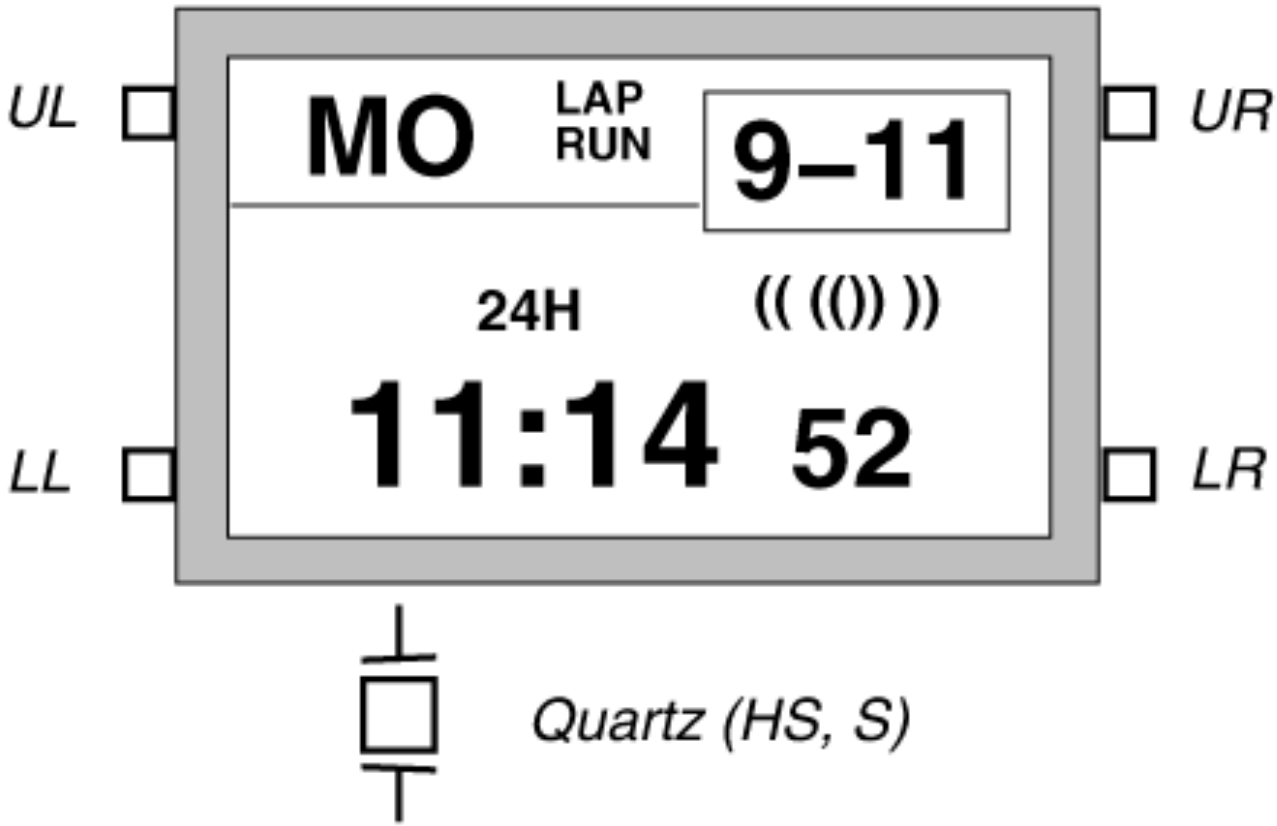


- 1985-6 : Esterel v2, premier vrai compilateur, première vraie distribution
  - résiduelle de Brzozowski : programmes → automates
  - GB, P. Couronné, Le_Lisp sur SM90 (!) puis Sun 3
  - soin particulier pour la bande de distribution et les exemples

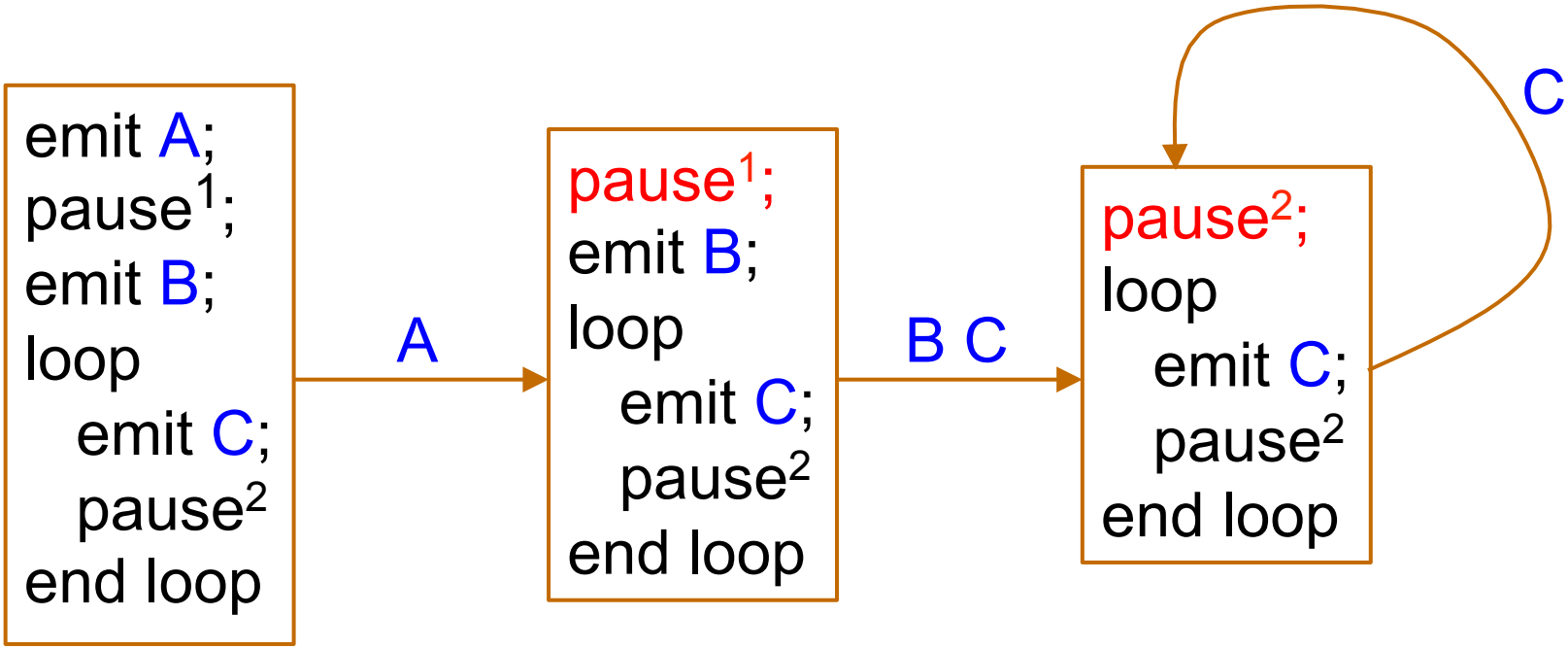
Faire directement en vrai, les protos c'est trop cher!



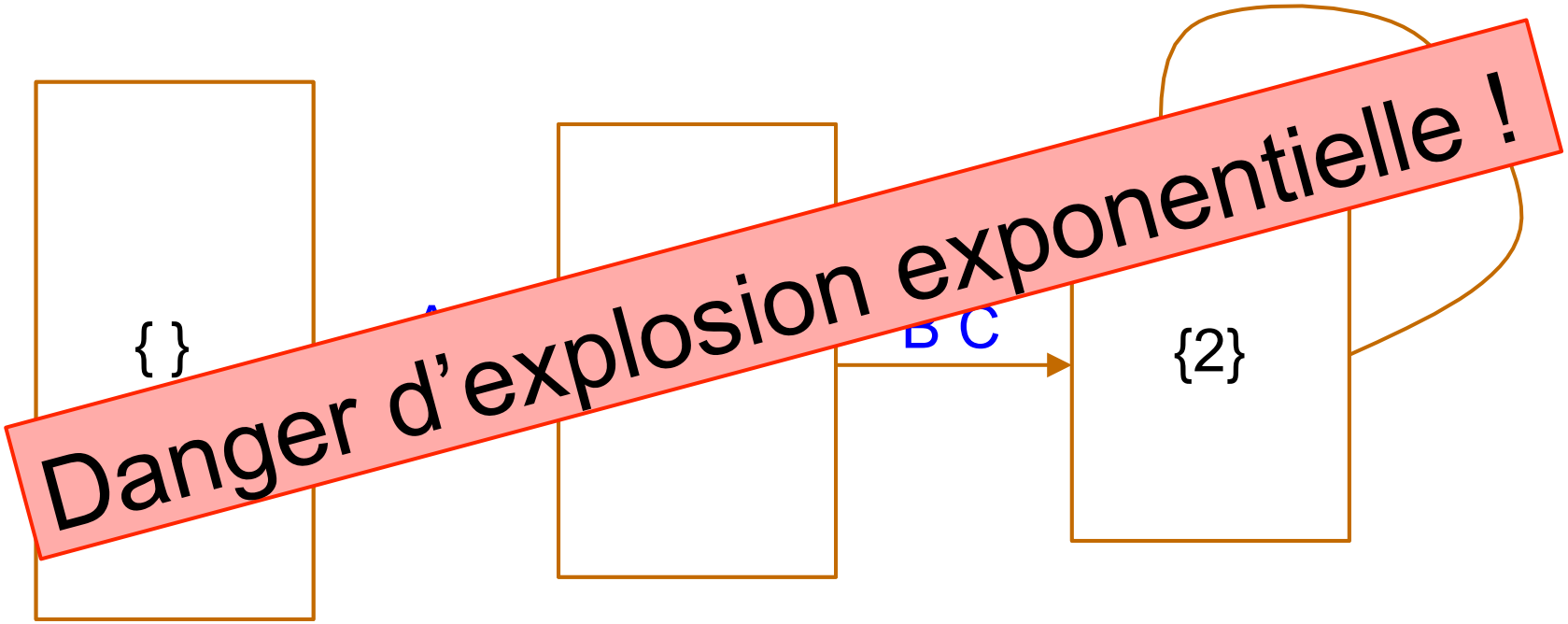
La montre digitale

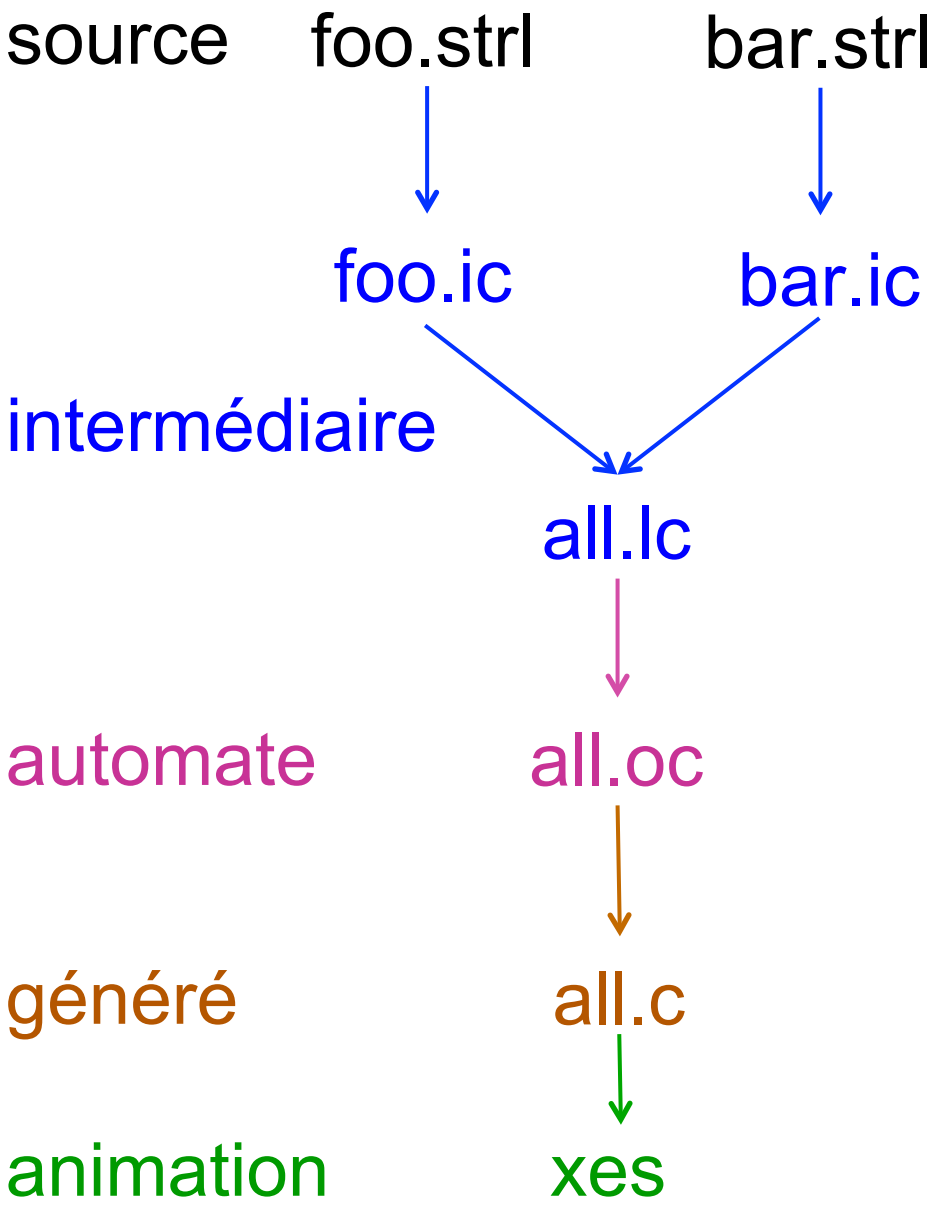


- 1986-9 : Esterel v3, premier compilateur sérieux, première industrialisation, premiers clients
  - algorithmes d'automates avec Ravi Sethi (Bell Labs)
  - thèse de G. Gonthier : codage efficace, causalité, réincarnation
  - mise en place d'une vraie équipe de développement avec une vraie stratégie logicielle



- 1986-9 : Esterel v3, premier compilateur sérieux, première industrialisation, premiers clients
  - algorithmes d'automates avec Ravi Sethi (Bell Labs)
  - thèse de G. Gonthier : codage efficace, causalité, réincarnation
  - mise en place d'une vraie équipe de développement avec une vraie stratégie logicielle





J-P Rigault, X. Fornari

R. Bernhard

JM. Tanzi

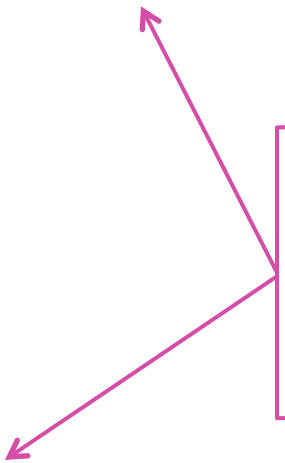
F. Boussinot

A. Ressouche

J-P. Marmorat

```
signals: 3
0: output: A 0 pure: previous: -  %lc: 2 8 0%
1: output: B 1 pure: previous: 0 %lc: 2 11 0%
2: output: C 2 pure: previous: 1 %lc: 2 14 0%
end:

statements: 7
0: Return: 0 %lc: 10 1 0%
1: Emit: [0] (2) %lc: 3 1 0%
2: Pause: (3) <0> 1 %lc: 4 1 0%
3: Emit: [1] (4) %lc: 5 1 0%
4: Emit: [2] (5) %lc: 7 3 0%
5: Pause: (6) <0> 2 %lc: 8 3 0%
6: Goloop: (4) %lc: 6 1 0%
end:
```



références
au code source
(backannotations)

Code commun avec Lustre

source foo.strl bar.strl

foo.ic bar.ic

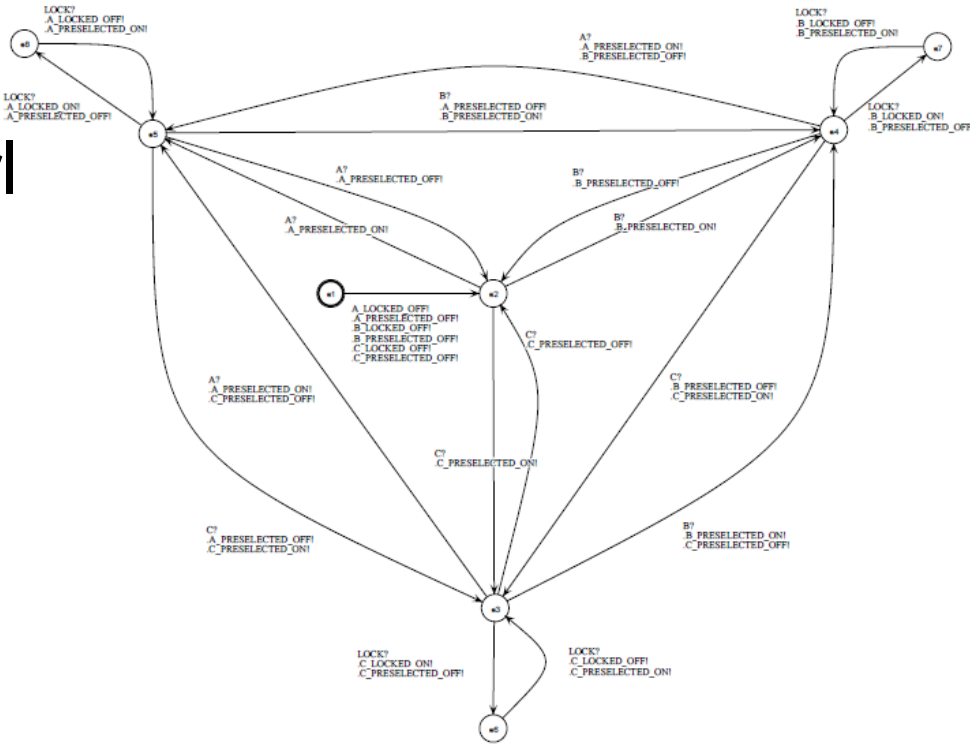
intermédiaire

all.lc

automate all.oc

all.c

généré



vérification formelle

Auto : R. de Simone
D. Vergamini

Autograph : V. Roy

Applis : V. Lecompte

Agenda

1. Programmer comme on pense
2. Donner un sens aux programmes
3. Implémenter et vérifier
4. Les utilisateurs pionniers
5. Les débuts industriels
6. Le choc électrique

Utilisateurs / collaborateurs pionniers

- Locaux :

  Centaur : G. Kahn, Yves Bertot

  IHM : Janet Incerpi (Bertot)

  Robotique : C. Borely, E. Coste-Manière, B. Espiau, D. Simon (ORRCAD)

  Astronomie : D. Gaffé

  Protocoles : C. Diot, C. Castellucia

  Automatismes, liaison OS : C. André, M-A. Peraldi

  Télécom : L. Arditi, C. André, *et. al.*

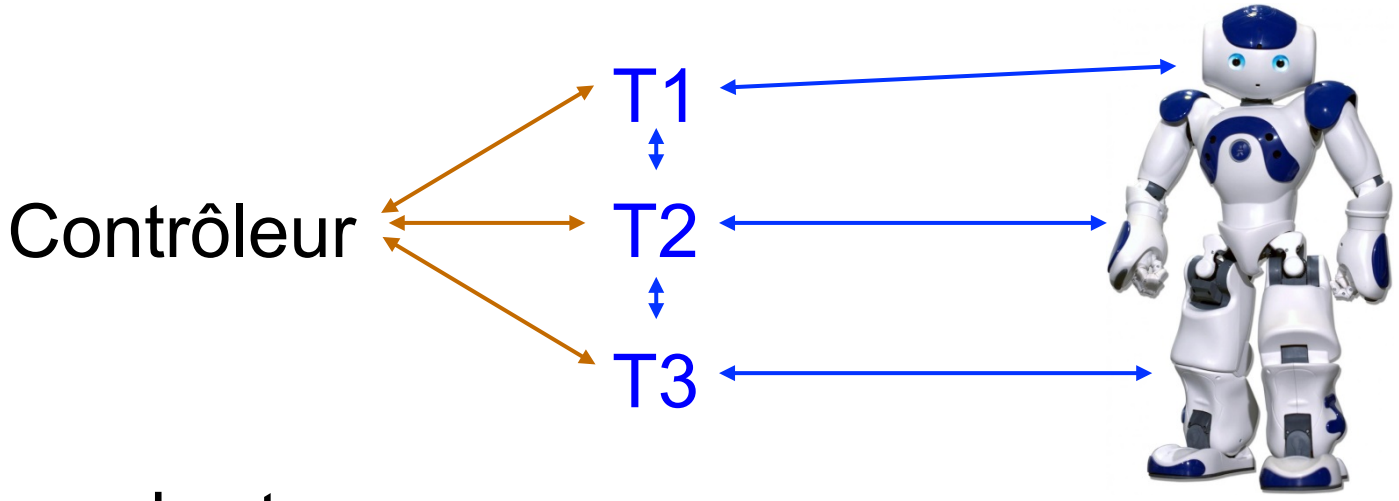
- Industrie

  Bell Labs, AT&T : R. Sethi, L. Jategaonkar, G. Gonthier

  Dassault Aviation : E. Ledinot, Y. Auffray, etc.

  Thomson, Bertin, Renault, IMRA, etc.

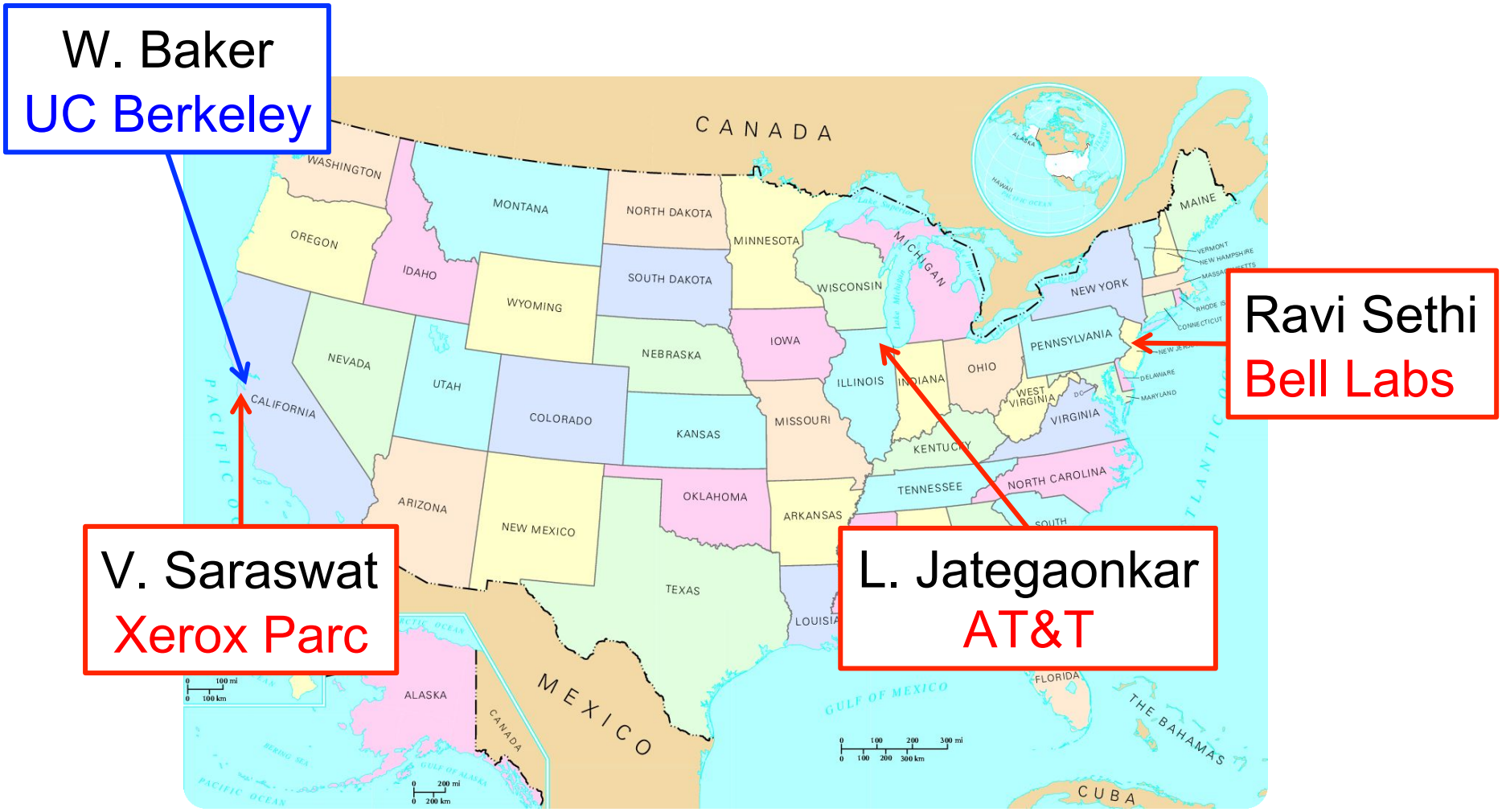
Séquencement de tâches asynchrones




```
abort
  { exec T1(...) || exec T2(...)} ;
   exec T3 (...)
when 30 Step
```

Avec préemption, traitement d'exceptions, etc.

J-P. Paris, E. Coste-Manière, équipe INRIA ORCADD


W. Baker
UC Berkeley
Ravi Sethi
Bell Labs
V. Saraswat
Xerox Parc
L. Jategaonkar
AT&T
CANADA
MEXICO
WASHINGTON
OREGON
CALIFORNIA
NEVADA
IDAHO
MONTANA
WYOMING
UTAH
ARIZONA
COLORADO
NEW MEXICO
NORTH DAKOTA
SOUTH DAKOTA
NEBRASKA
KANSAS
OKLAHOMA
TEXAS
MINNESOTA
IOWA
MISSOURI
ARKANSAS
LOUISIANA
WISCONSIN
ILLINOIS
MICHIGAN
INDIANA
OHIO
KENTUCKY
TENNESSEE
MISSISSIPPI
ALABAMA
GEORGIA
FLORIDA
SOUTH CAROLINA
NORTH CAROLINA
VIRGINIA
WEST VIRGINIA
PENNSYLVANIA
NEW YORK
MAINE
VERMONT
NEW HAMPSHIRE
MASSACHUSETTS
RHODE ISLAND
CONNECTICUT
DELAWARE
MARYLAND
ALASKA
HAWAII
GULF OF MEXICO
THE BAHAMAS
CUBA

Esterel en Asie

Réaction du monde académique ?

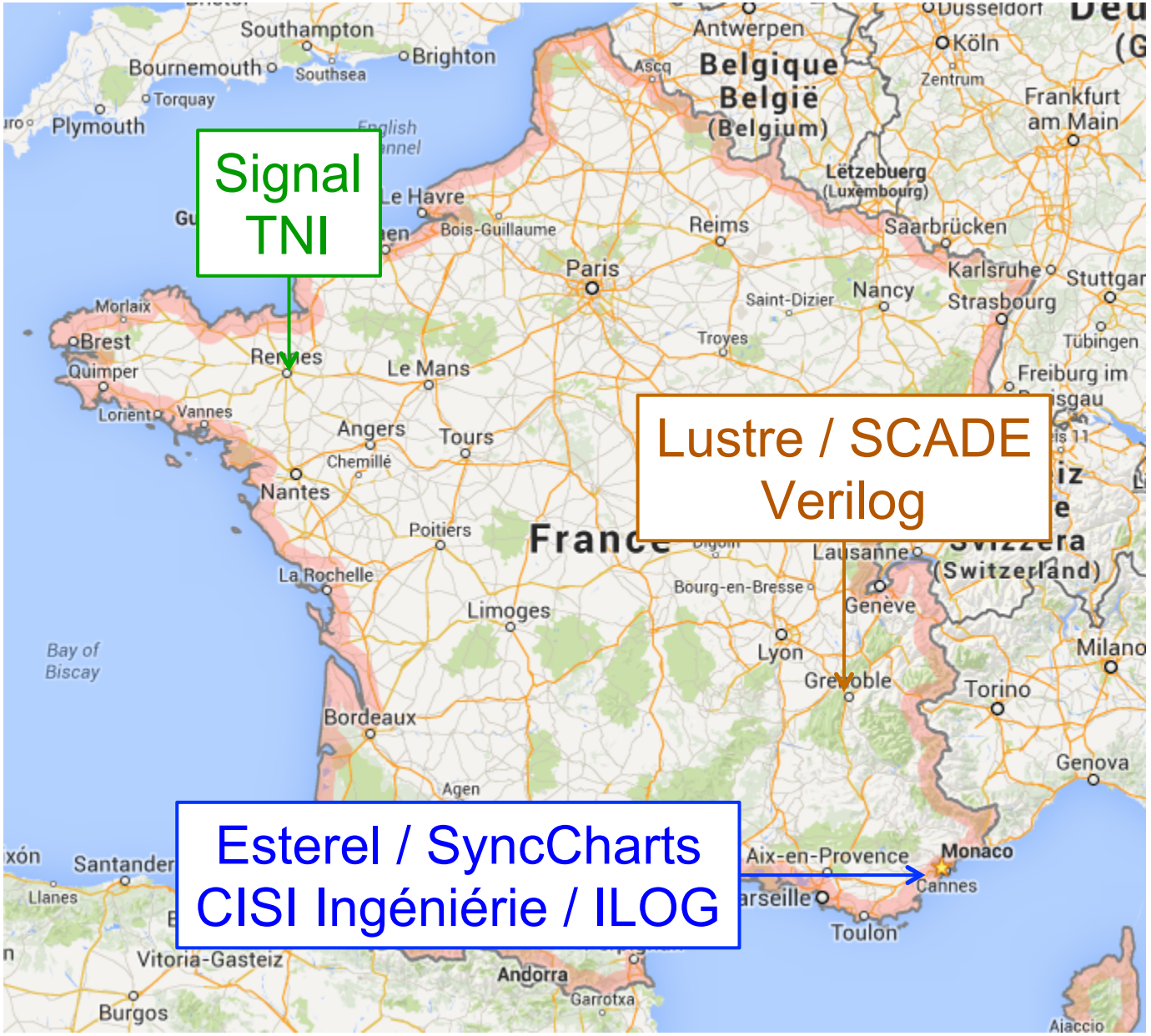
- Intérêt globalement mitigé – quelques groupes actifs
  - dictature de facto de l'asynchronisme,
    presque partout jugé synonyme du parallélisme ☹

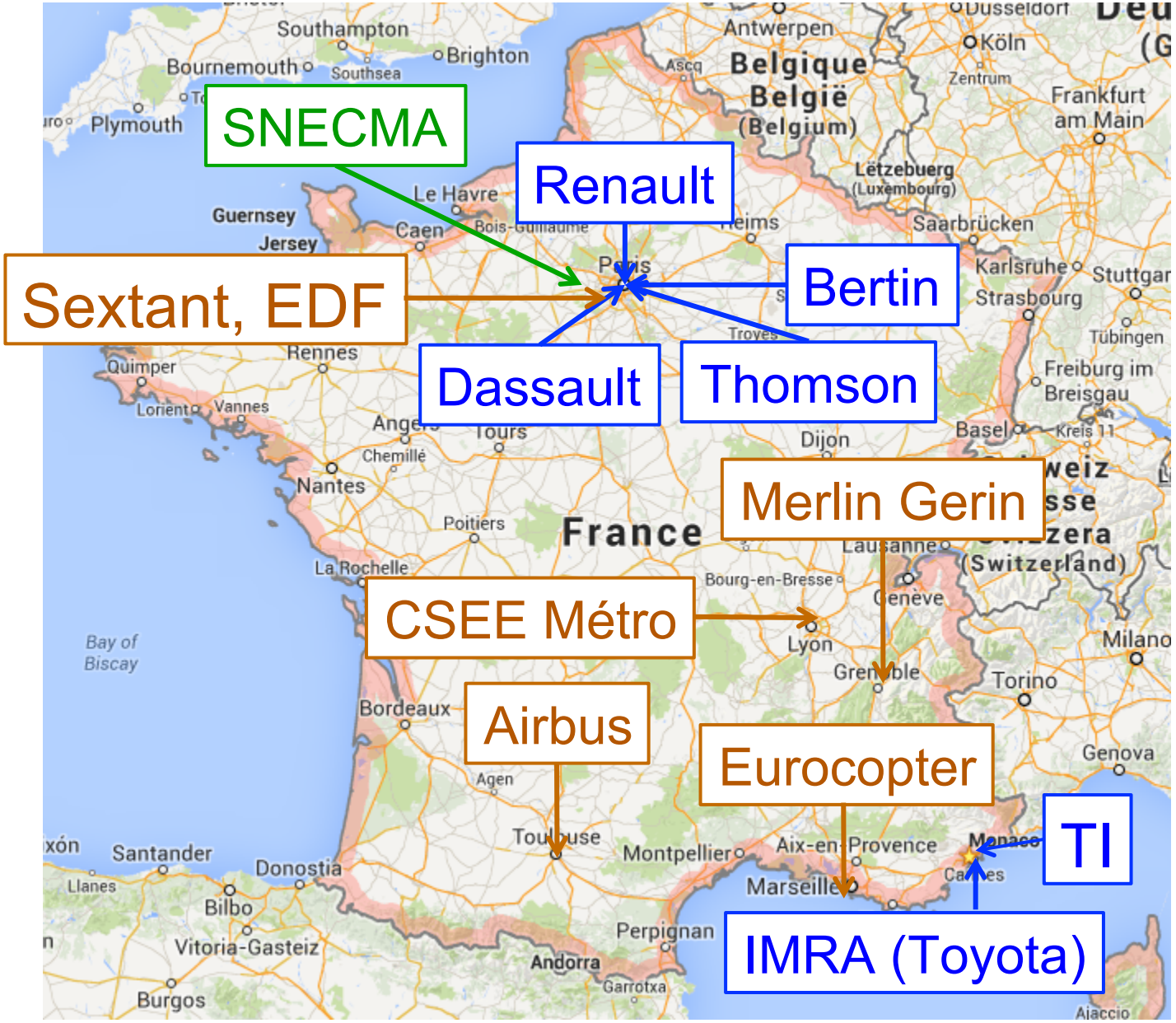
- Même réaction pour les Statecharts de David Harel
  - article rejeté par plusieurs journaux
  - maintenant cité plus de 7650 fois...

En recherche, les idées non-standards
ne sont pas vraiment les bienvenues !
(au moins au début)

Agenda

1. Programmer comme on pense
2. Donner un sens aux programmes
3. Implémenter et vérifier
4. Les utilisateurs pionniers
5. Les débuts industriels
6. Le choc électrique


Signal
TNI
Lustre / SCADE
Verilog
Esterel / SyncCharts
CISI Ingéniérie / ILOG
Southampton
Brighton
Bournemouth
Southsea
Torquay
Plymouth
English Channel
Antwerpen
Belgique
België
(Belgium)
Ascq
Köln
Zentrum
Frankfurt am Main
Lëtzebuerg
(Luxembourg)
Le Havre
Bois-Guillaume
Reims
Saarbrücken
Paris
Karlsruhe
Stuttgart
Nancy
Saint-Dizier
Strasbourg
Morlaix
Brest
Quimper
Rennes
Troyes
Tübingen
Le Mans
Freiburg im
Lorient
Vannes
Angers
Tours
Chemillé
Nantes
Poitiers
France
Lausanne
Switzerland
La Rochelle
Bourg-en-Bresse
Genève
Limoges
Lyon
Milano
Bay of Biscay
Grenoble
Torino
Bordeaux
Genova
Agen
Santander
Aix-en-Provence
Monaco
Cannes
Marseille
Toulon
Vitoria-Gasteiz
Andorra
Garrotxa
Burgos
Ajaccio


SNECMA
Renault
Sextant, EDF
Bertin
Dassault
Thomson
Merlin Gerin
CSEE Métro
Airbus
Eurocopter
TI
IMRA (Toyota)
Southampton
Brighton
Bournemouth
Plymouth
Antwerpen
Belgique
België
(Belgium)
Köln
Frankfurt am Main
Lëtzebuerg (Luxembourg)
Le Havre
Guernsey
Jersey
Caen
Saarbrücken
Karlsruhe
Strasbourg
Stuttgart
Tübingen
Freiburg im Breisgau
Quimper
Rennes
Lorient
Vannes
Chemillé
Nantes
Tours
Dijon
Basel
Poitiers
France
La Rochelle
Lausanne
(Switzerland)
Bourg-en-Bresse
Genève
Lyon
Milano
Torino
Bay of Biscay
Bordeaux
Agen
Genova
Toulouse
Montpellier
Aix-en-Provence
Marseille
Monaco
Santander
Donostia
Llanes
Bilbo
Vitoria-Gasteiz
Burgos
Perpignan
Andorra
Garrotxa
Ajaccio

Rafale = Esterel, Airbus = SCADE

Solution 1 : combat aérien

Solution 2 : unification des langages

→ Esterel v7, SCADE 6

Agenda

1. Programmer comme on pense
2. Donner un sens aux programmes
3. Implémenter et vérifier
4. Les utilisateurs pionniers
5. Les débuts industriels
6. Le choc électrique

1990 : Le choc électrique

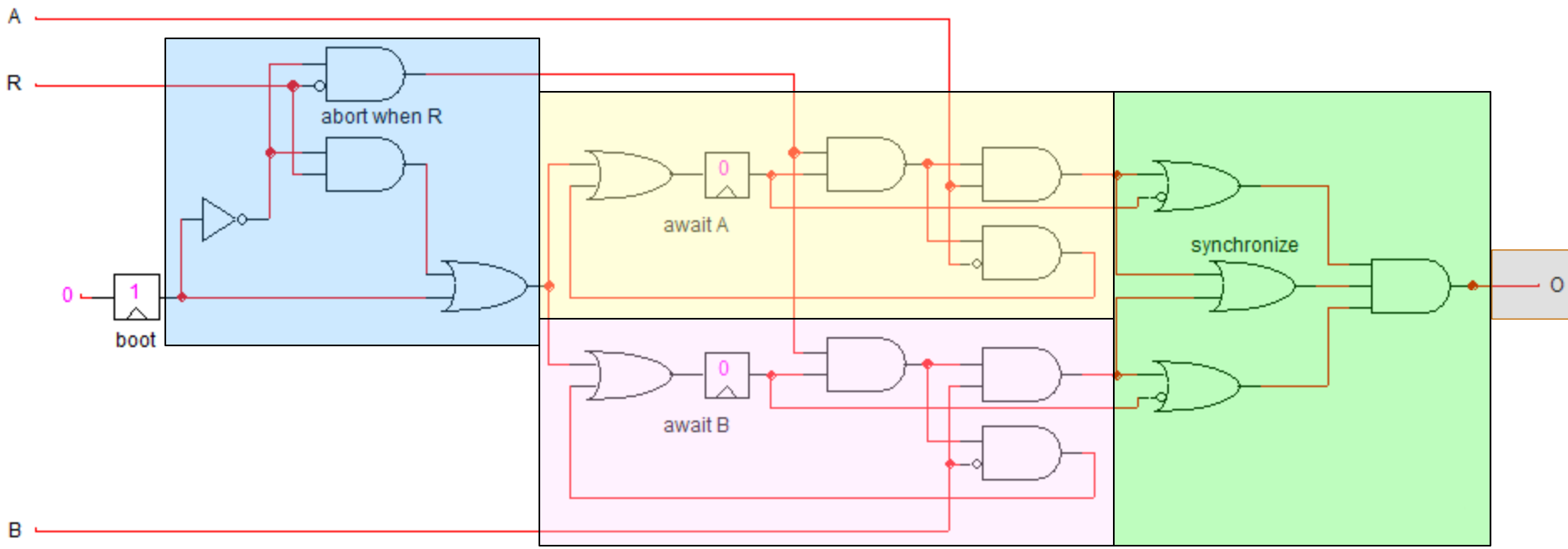
Equipe de J. Vuillemin chez Digital Equipment PRL
coprocesseur FPGA Perle 1

- Traduction directe en circuit digitaux
  Génération de logiciel par simulation du circuit

Danger d'explosion exponentielle !

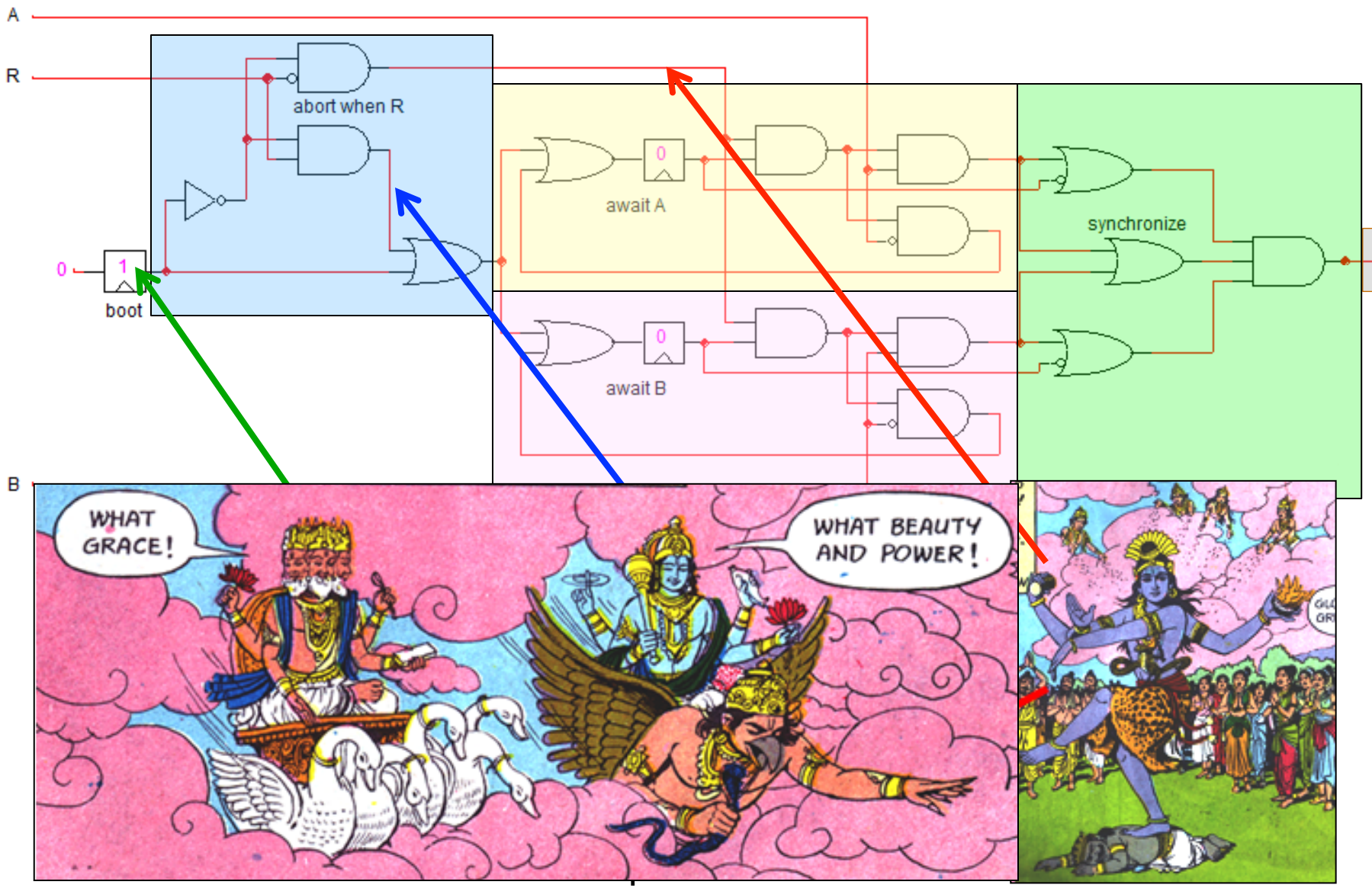
- Calculs sur les circuits à l'aide des BDDs (tous neufs)
  - optimisation matérielle et logicielle, vérification formelle

ABRO en circuits

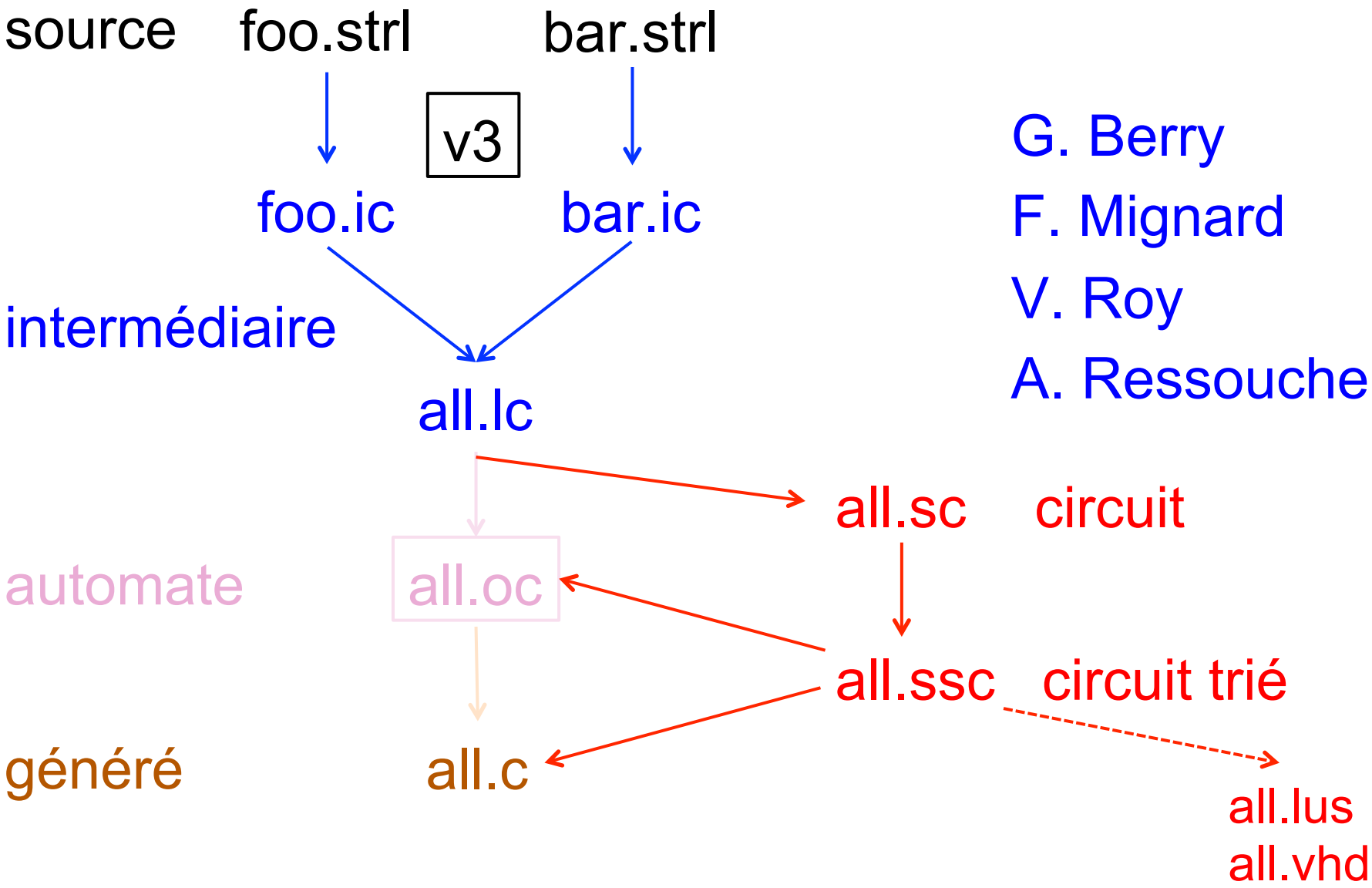



```
loop
  abort
      { await A || await B };
    emit O ;
     halt
  when R;
end loop
```

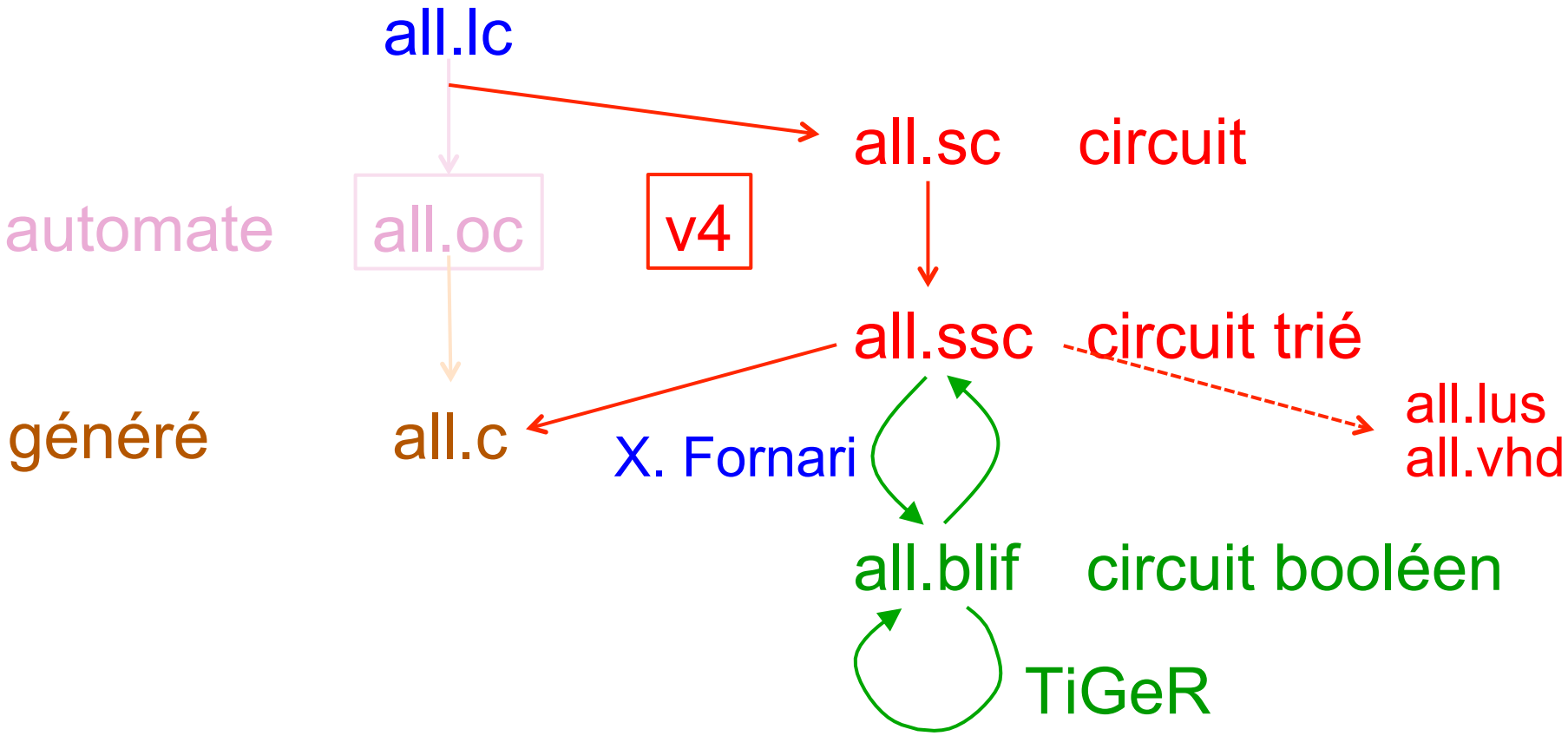
ABRO en circuits



Compilateur esterel v4



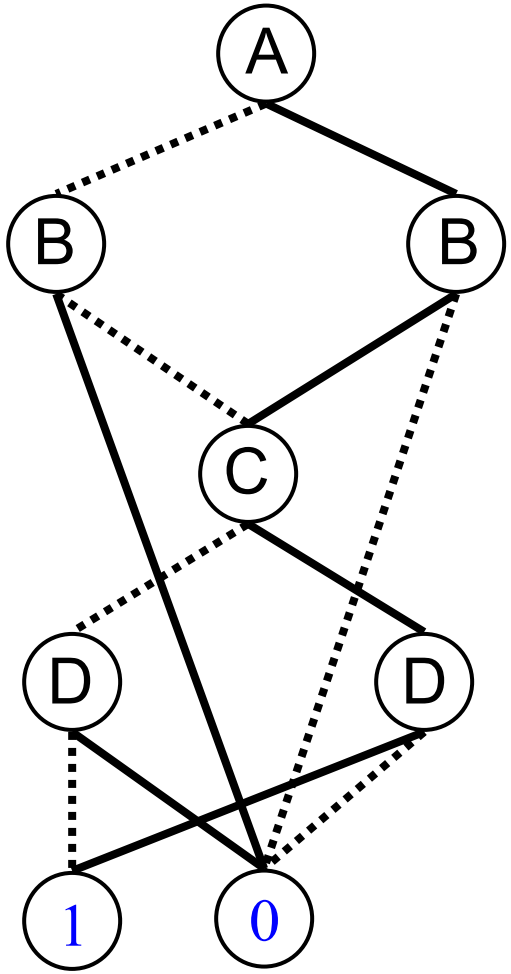
Optimisation / vérification formelle



blif = Berkeley Logic Interchange Format

Binary Decision Diagrams (BDDs)

$(A \Leftrightarrow B) \wedge (C \Leftrightarrow D)$



R. Bryant (CMU), ?. JP Billon (Bull)
lutte contre l'exponentielle

O. Coudert, J-C. Madre, H. Touati
(Digital Equipment PRL)
implémentation efficace
algorithmes image directe / inverse

Autres applications :
circuits, traitement d'images,
géométrie algorithmique, etc.

Exemple de vérification formelle

Dassault : BTM = Boîtier de Test et Maintenance
tests au sol et tests en vol

Implémentation classique : OS + tâches asynchrones

Prouver : les tests au sol ne peuvent être lancés en vol
exemple : faire flapper tous les ailerons

Idée : ~~tasking asynchrone non-déterministe~~
→ programme Esterel déterministe
+ observateurs synchrones pour la vérif (Lustre)

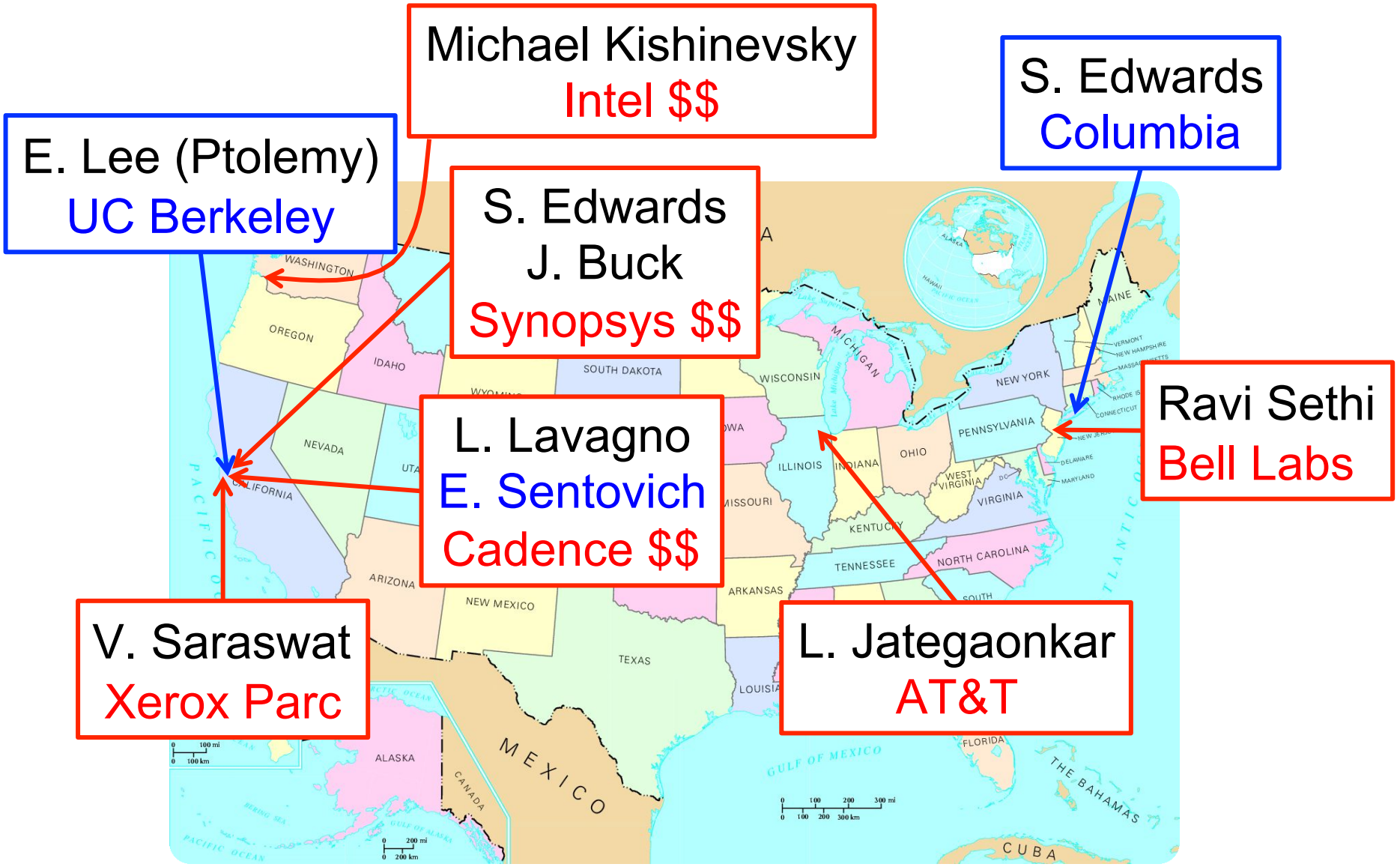
Gros programme, grosse preuve, mais faisable!

Merci à E. Ledinot, E. Nassor *et. al*.

Compilation modulaire : Esterel v6

Mais réussi chez Dassault dans un cadre plus restreint

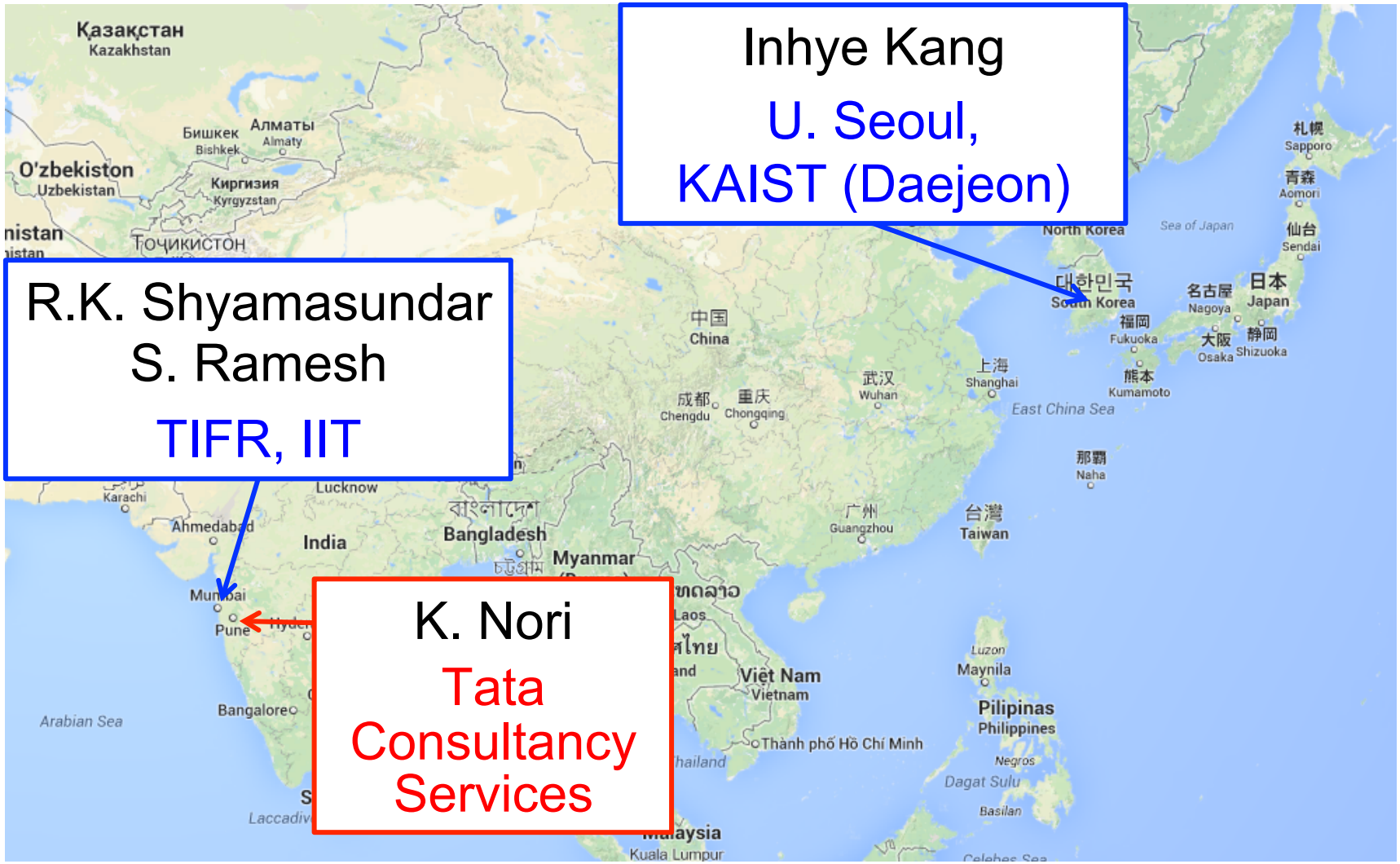
E. Nassor, O. Hainque, *et. al.*


Michael Kishinevsky
Intel $$
S. Edwards
Columbia
E. Lee (Ptolemy)
UC Berkeley
S. Edwards
J. Buck
Synopsys $$
Ravi Sethi
Bell Labs
L. Lavagno
E. Sentovich
Cadence $$
V. Saraswat
Xerox Parc
L. Jategaonkar
AT&T
WASHINGTON
OREGON
IDAHO
NEVADA
CALIFORNIA
ARIZONA
NEW MEXICO
SOUTH DAKOTA
WISCONSIN
MICHIGAN
ILLINOIS
INDIANA
OHIO
PENNSYLVANIA
NEW YORK
MAINE
WEST VIRGINIA
VIRGINIA
KENTUCKY
TENNESSEE
NORTH CAROLINA
ARKANSAS
TEXAS
LOUISIANA
FLORIDA
THE BAHAMAS
CUBA
MEXICO
ALASKA
CANADA
GULF OF MEXICO

Que pensez-vous des Français?

Beaucoup de bien : ils sont forts en maths,
bons programmeurs, bosseurs
et ce ne sont pas des concurrents !

Esterel en Asie



Pendant l'industrie, la recherche continue

- Allemagne
  - Quartz / Averest : K. Schneider, U. Kaiserslautern
  - Lego Mindstorms : R. van Hanxleden, U. Kiel
  - SCL : Sequential consistency language
    - J. Aguado, R, van Hanxleden, M. Mendler, I. Fuhrman, *et. al.*
- Suisse : L. Zaffalon, EIG Genève
  - *Programmation synchrone de systèmes réactifs avec Esterel et les SyncCharts*
  - Presses polytechniques et universitaires Romandes, 2005
- Corée : Inhye Kang *et. al.*, U. Séoul
  - *Esterel programming for beginners*
  - www.hongpub.co.kr (2005)
- Inria Sophia : M. Serrano, GB, C. Nicolas
  - *Hop and HipHop: Multitier Web Orchestration*
  - ICDCIT conf., Bhubaneswar, février 2014

B. Serpette, autres sémantique / compilation CPS

Enseignements de l'aventure

- Aucune des étapes clefs n'était « programmée »
  - le développement se programme, pas la découverte !
  - mais il ne suffit pas de ne pas programmer pour réussir...
- Aucun des vrais progrès n'a été fait de l'intérieur
  - tous l'ont été par des contacts imprévus et variés
  - automatique, automates, circuits, BDDs, scheduling, analyse statique, programmation fonctionnelle, etc.
- Le rôle des collaborations industrielles a été fondamental
  - être son propre utilisateur est insuffisant, car trop facile
  - la taille des vraies applications est considérable
  - les clients sont plus concernés par les défauts des langages et outils que par leurs qualités

- Assurer la cohésion de l'équipe est indispensable
  - mais les points de vue et engagements sont très variables
  - et faire de gros projets logiciels avec des thésards est risqué
- Toute question sérieuse doit provoquer une réunion
  - interdiction de reconvertir des salles de réunion !
- La gestion logicielle doit être géniale (merci Xavier)
  - rien ne diverge aussi vite qu'un logiciel !
- La sémantique mathématique est le seul vrai guide

Voleriez-vous dans un avion piloté par vos propres programmes?