# Parallélisme asynchrone

Gérard Berry

Collège de France
Chaire Informatique et sciences numériques

Cours 5 du 6 janvier 2010

# *Agenda*

1. Pourquoi le parallélisme
2. De nouvelles questions
3. Extensions de la programmation séquentielle
4. Réseaux graphiques
5. Calculs de processus et modèles chimiques
6. Le pi-calcul et ses variantes

# *Agenda*

1. Pourquoi le parallélisme
2. De nouvelles questions
3. Extensions de la programmation séquentielle
4. Réseaux graphiques
5. Calculs de processus et modèles chimiques
6. Le pi-calcul et ses variantes

# *Gérer les tâches parallèles*

- Exécuter plusieurs tâches sur un seul processeur
  - assurer et orchestrer leurs accès aux ressources (mémoires, disques, imprimantes, etc.)
  - régler les compétitions

  $\Rightarrow$ systèmes d'exploitation, ordonnancement (scheduling)

- Répartir une application sur plusieurs processeurs
  - calcul numérique (éléments finis), grandes simulations, etc.
  - graphique 3D, imagerie médicale, traitement du signal

  $\Rightarrow$ systèmes et intergiciels de machines parallèles

# *Optimiser*

- Accélérer l'exécution matérielle des instructions
  - pipeline, cache, spéculation
  - accélérateurs matériels de fonctions précises

  $\Rightarrow$ architectures RISC, mémoire virtuelle, etc.

- Augmenter le nombre d'unités de calcul
  - ordinateurs vectoriels / massivement parallèles, grilles
  - SIMD, MIMD, multicoeurs

  $\Rightarrow$ compiler pour ces architectures

# *Communiquer*

- Transmettre de l'information entre composants
  - ordinateurs, imprimantes, modems, écrans
  - transmission série (USB, Firewire) ou parallèle (HDMI)
  - réseaux sur chips
- Connecter des ordinateurs ou équipements proches
  - réseaux locaux (Ethernet, réseaux jetons, TTP, WiFi)
  - réseaux cellulaires (GSM, 3G, etc.)

  $\Rightarrow$ protocoles de communication, émetteurs, récepteurs
- Connecter des équipements lointains
  - réseau téléphonique, satellites
  - Internet

  $\Rightarrow$ TCP / IP

# *Interfacer*

- Interagir physiquement
  - souris, clavier, joystick
  - manettes accélérométriques (wii)
  - écrans, haut-parleurs, etc.
- Visualiser
  - applications individuelles
  - fenêtres de navigateur
  - écrans multiples dans un cockpit

# *Partager / chercher*

- Partager de l'information à grande échelle
  - Wikipedia, OpenStreetMap
  - réseaux sociaux
  - diffusion de musique et de film

  $\Rightarrow$ serveurs de données, pair à pair

- Chercher de l'information à grande échelle
  - moteurs de recherches généraux
  - recherche dans des communautés spécifiques
  - serveur centralisé $\rightarrow$ pair à pair

# *Contrôler*

- Contrôler des systèmes compacts
  - gérer capteurs et actionneurs
  - synchroniser le son et l'image, les 3 axes d'un avion, les 4 freins d'une voiture, les articulations d'un robot
- Coordonner des grands systèmes
  - contrôle aérien, aiguillages de train
  - circulation dans les villes
  - routage Internet
- Rendre les systèmes robustes aux pannes
  - redondance d'équipements et de logiciels

# *Agenda*

1. Pourquoi le parallélisme
2. De nouvelles questions
3. Extensions de la programmation séquentielle
4. Réseaux graphiques
5. Calculs de processus et modèles chimiques
6. Le pi-calcul et ses variantes

# *De nouvelles questions*

- Asynchronisme / synchronisme / vibration
- Modélisation / programmation
- Déterminisme / non-déterminisme
- Architecture statique / dynamique
- Centralisation / distribution
- Distribution logique / géographique
- Mobilité des données, migration des calculs
- Sécurité des données et des échanges

Qu'est ce que le comportement
d'un système parallèle ?

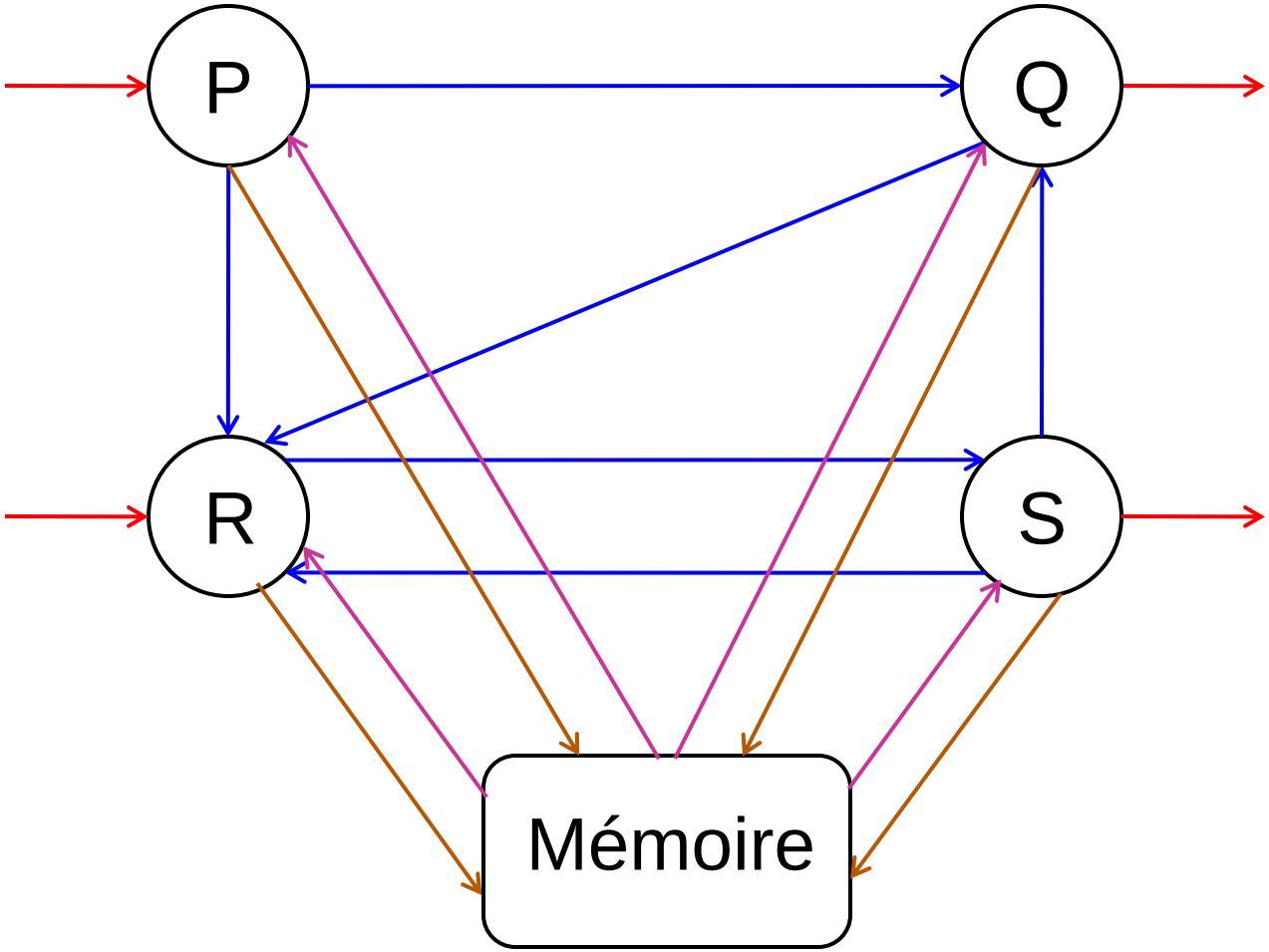
# *La transformation du calcul*

- Perte de la notion simple de fonction
  - l'arrêt est souvent non souhaitable
  - le non déterminisme est souvent la règle
- Parallélisme asynchrone => nouveaux bugs
  - interblocage (deadlock)
  - famine (starvation)
  - course
  - interférence
- Débogage distribué très difficile
  - comment retrouver la causalité des événements?

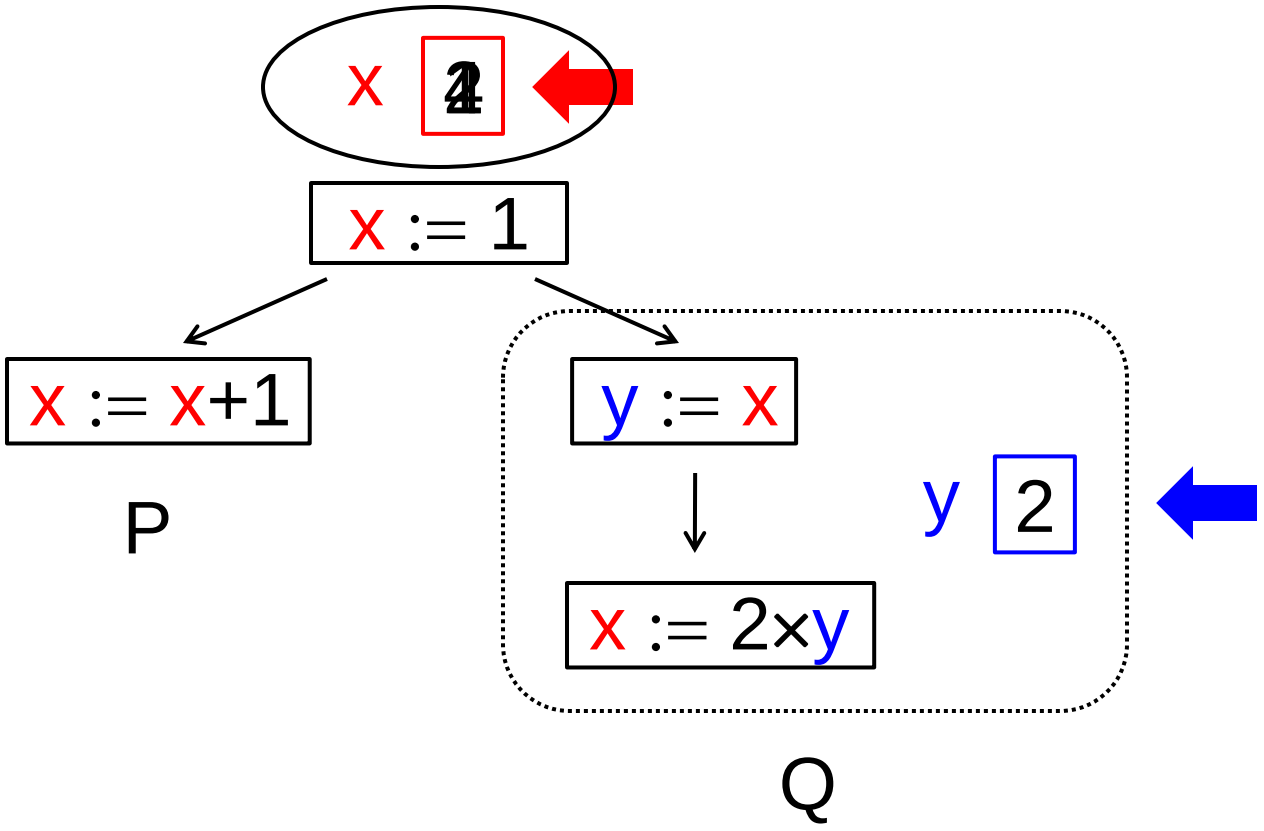
# *Agenda*

1. Pourquoi le parallélisme
2. De nouvelles questions
3. Extensions de la programmation séquentielle
4. Réseaux graphiques
5. Calculs de processus et modèles chimiques
6. Le pi-calcul et ses variantes

# *Mémoire partagée*

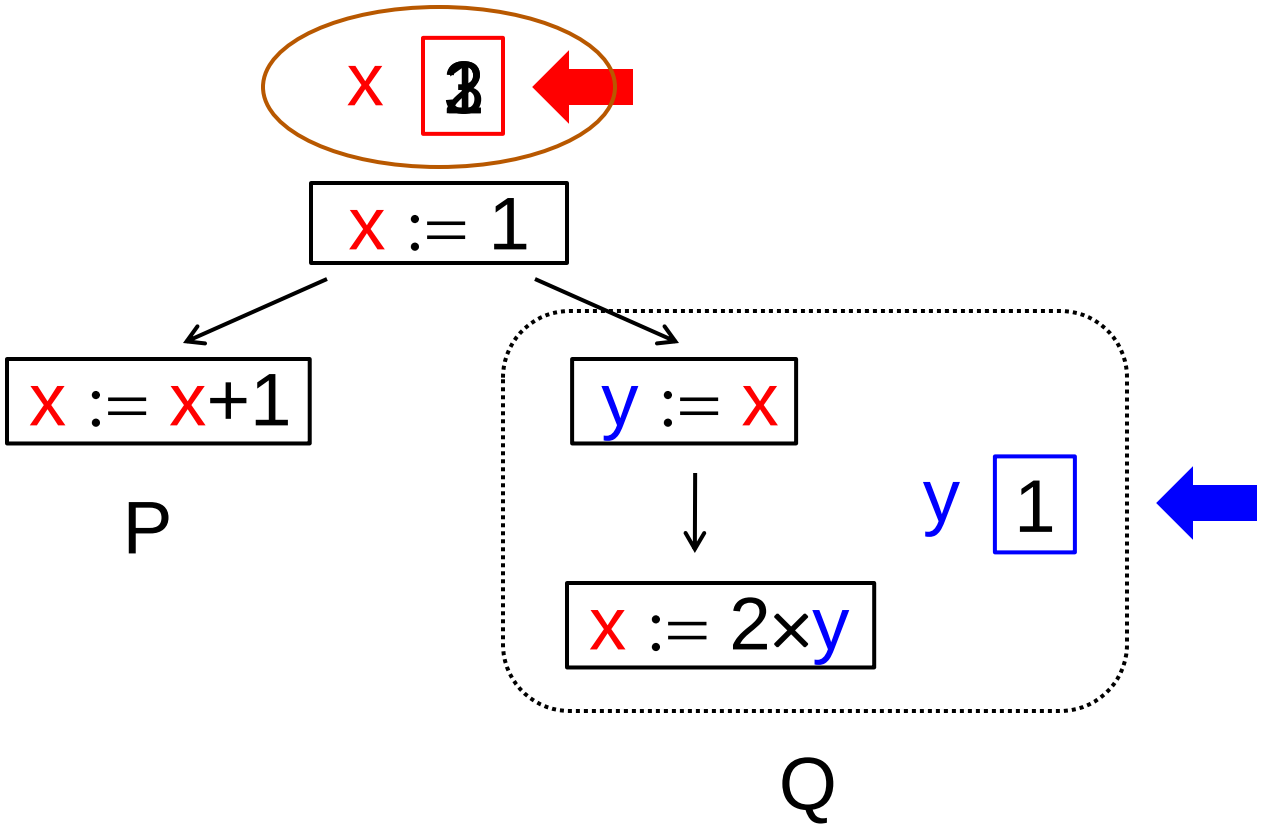


# *Threads et mémoire partagée*



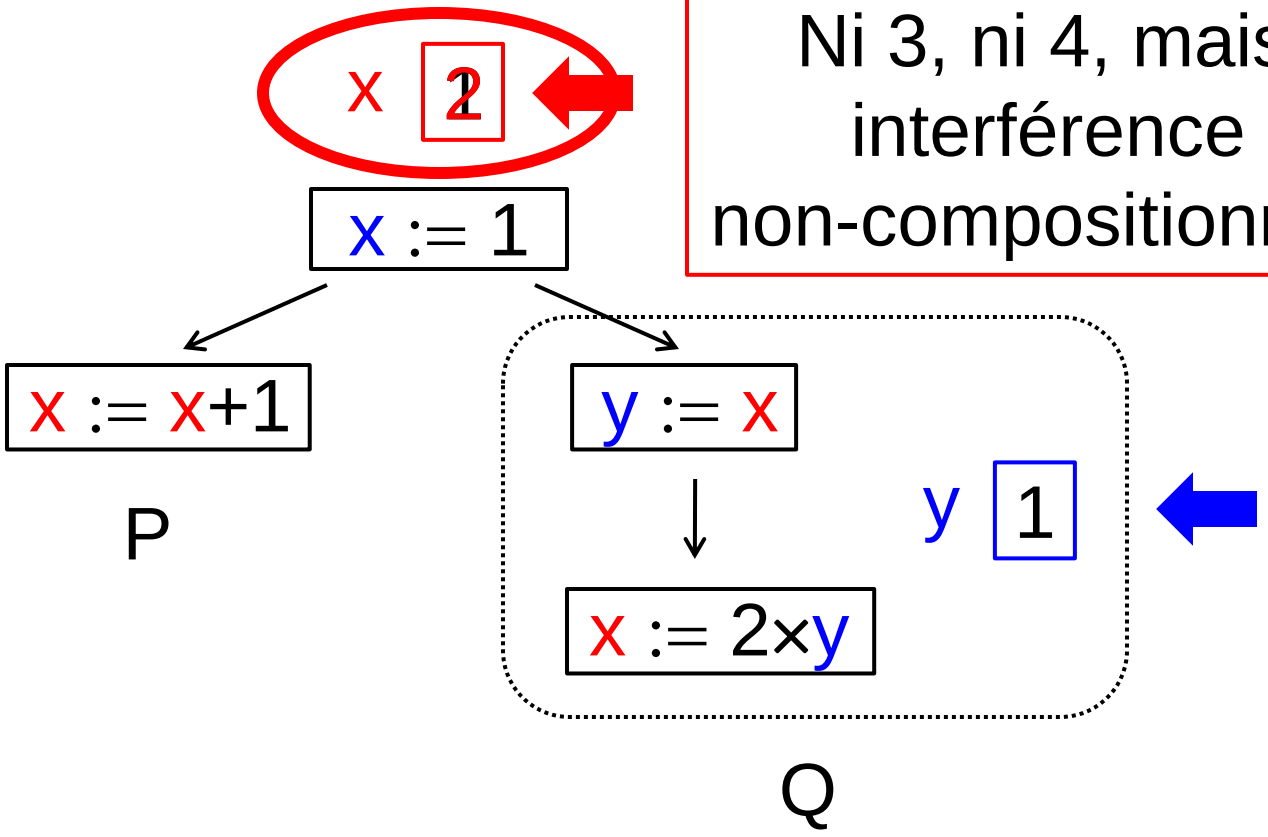

1. P || Q exécuté comme P; Q

# *Threads et mémoire partagée*




2. P || Q exécuté comme Q; P

# *Threads et mémoire partagée*

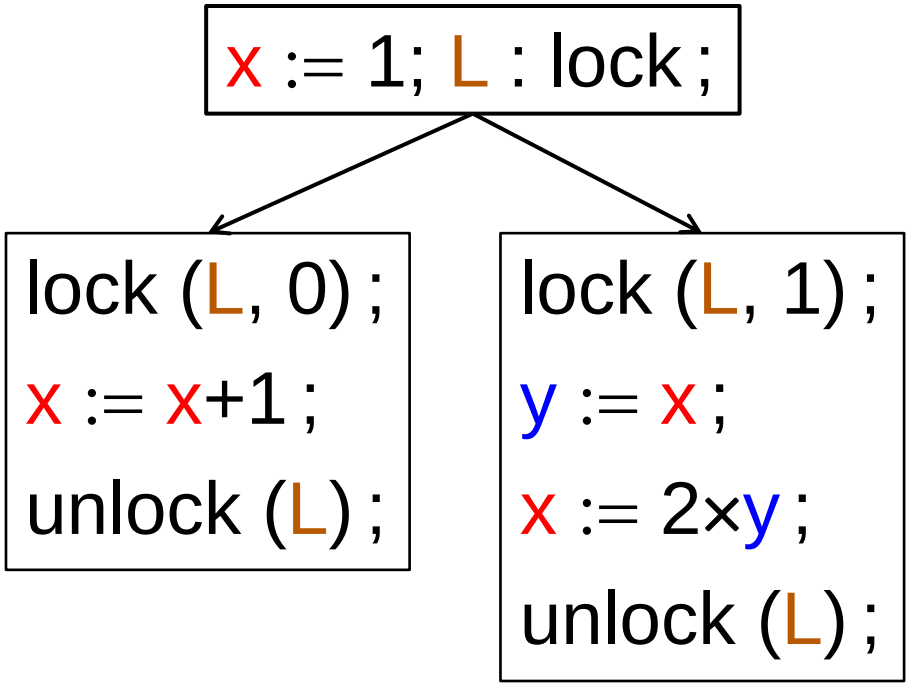



Ni 3, ni 4, mais 2 !
interférence ⇒
non-compositionnalité !

3. P || Q complètement entrelacés

# *Exclusion mutuelle*

- Assurer que « x := x+1 » et « y := x ; x := 2×y » sont exécutés de façon exclusive (ou atomique)
- Méthode : implémenter un verrou partagé, délimiter des sections critiques

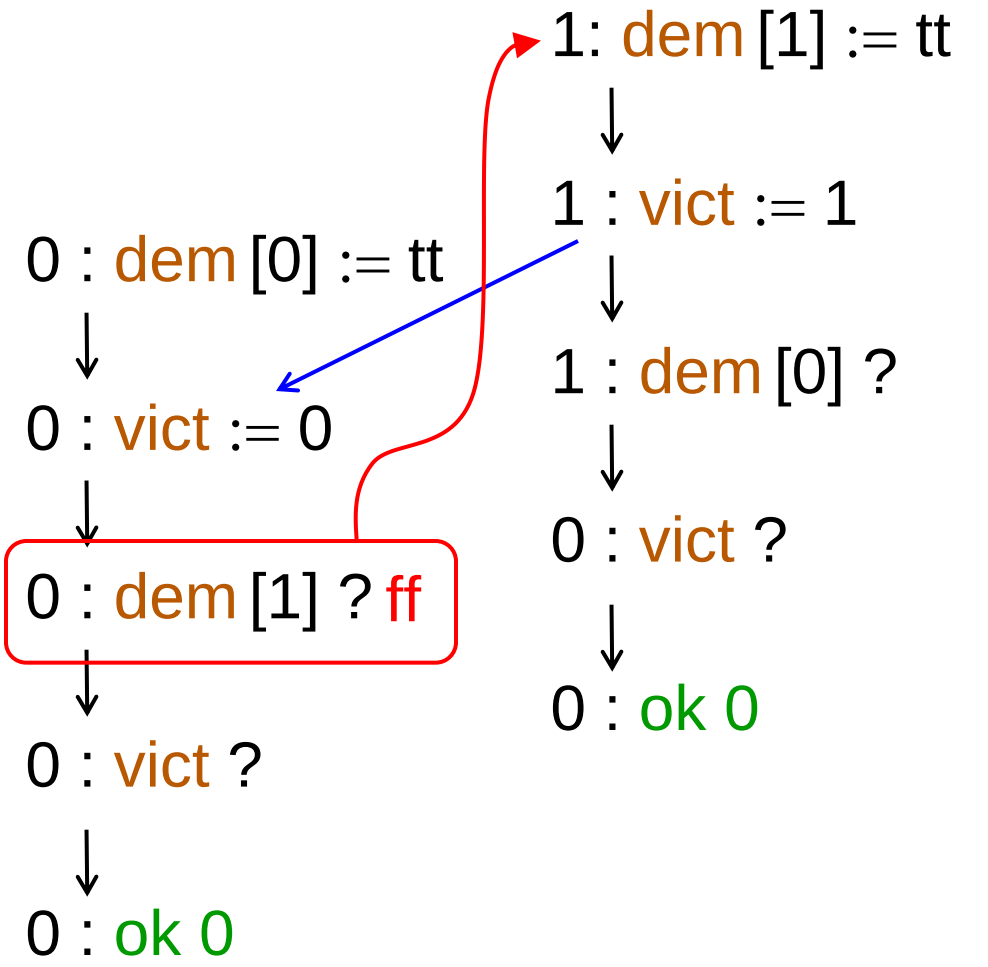



- verrou à 2 : TestAndSet
  Peterson
- verrou à n : Boulangerie
  L. Lamport

# *Algorithme de Peterson*

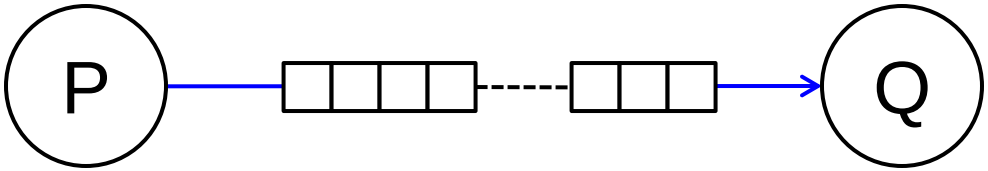
partagés | bool demande[2] ;
bool victime ;

```
lock(int id) {
   int autre = 1 – id ;
   demande [id] = true ;
   victime = id ;
   while (demande [autre]
            && victime == id)
        {}  // attente
   // ok
}
unlock(int id) {
   demande [id] = false ;
}
```



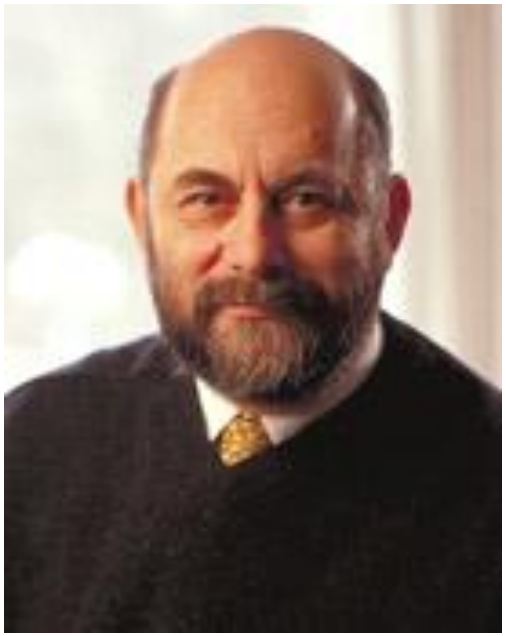

Le temps n'est pas cyclique !

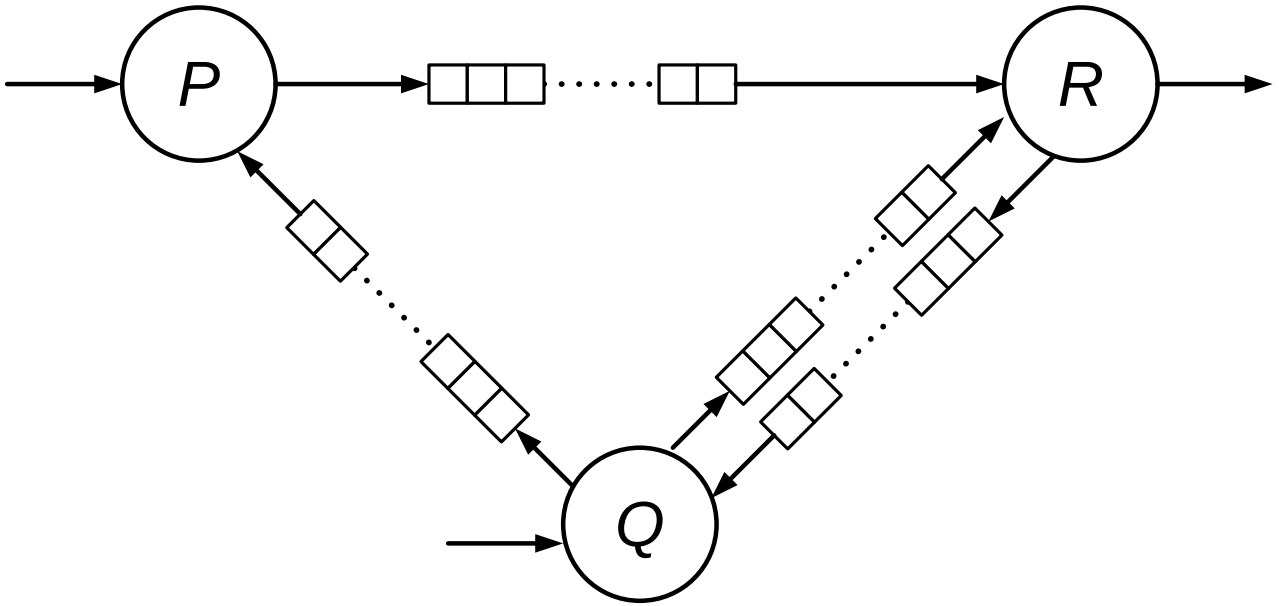
# *Communication par files d'attente*




- Découplage de l'émetteur et du récepteur
  - parallélisme évident, bien adapté à la distribution
  - mais introduction de latence imprévisible
  - et comportement total plus complexe
- Applications
  - protocoles de transmission dans les réseaux
  - réseaux sur puce
  - traitement du signal
  - interfaces homme / machine, Web

# *Réseaux de Kahn*

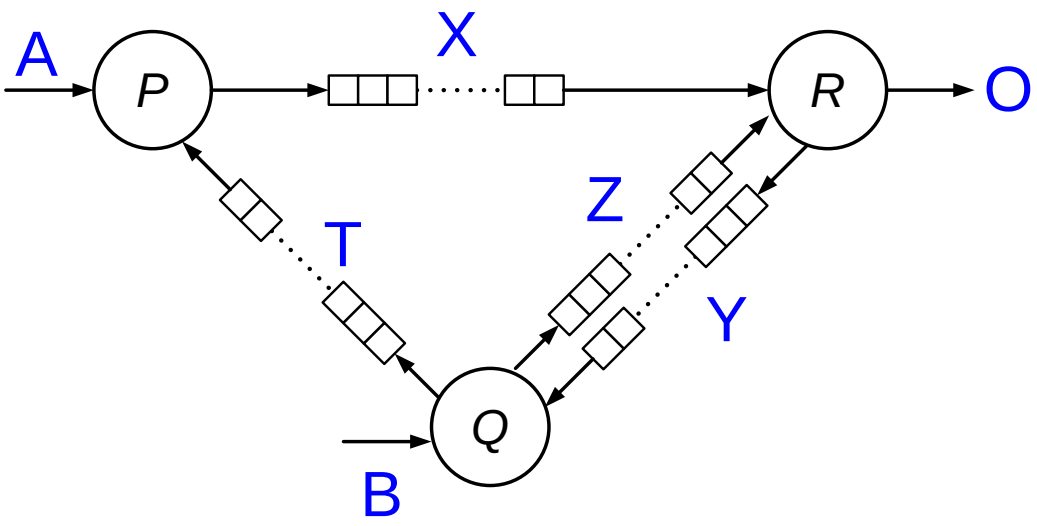





- nœuds séquentiels, ou récursivement réseaux de Kahn
- get bloquant, send non-bloquant

  $\Rightarrow$ files FIFO non bornées
- ordonnancement arbitraire des calculs

# *Sémantique par points fixes de Scott*




flots finis ou infinis

$X = x_0, x_1, \ldots, x_n, \ldots$

ordre préfixe $\rightarrow$ cpo

$x_0, \ldots, x_n \subset x_0, \ldots, x_n, \ldots$

P, Q, R : fonctions continues de flots

$X = P(A,T)$

$(O,Y) = R(X, Z)$

$T = Q(B,Y)$

Etant donnés, A, B

X, Y, Z = plus petits (i.e. plus courts)

points fixes du système d'équation

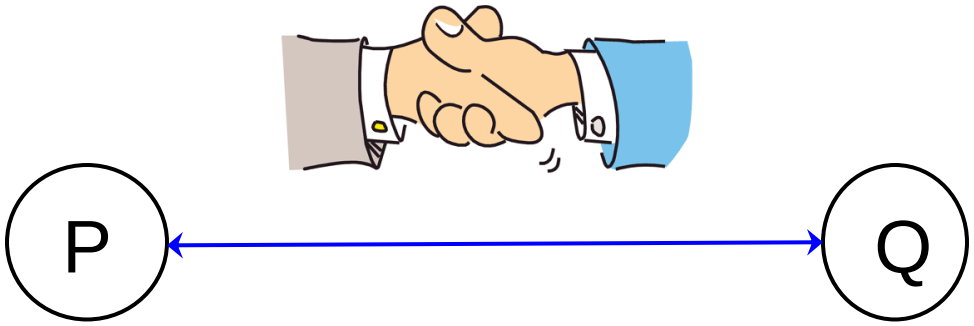
=> comportement global déterministe

# *Influence des réseaux de Kahn*

- Modèles flots de données asynchrones
  - Ptolemy (Ed Lee, Berkeley)
  - Lucid (Aschroft / Wadge, Waterloo)
  - audio, vidéo, traitement du signal (industrie)
- Modèles flots de données synchrones
  - Lustre, Signal : contraintes de type => FIFOs de taille 0
  - Réseaux N-synchrones => FIFOs de taille bornée
  - Réseaux de Conal Elliott : animation, etc
- Utilisation de flots infinis dans d'autres domaines
  - lazy-lists, coinduction, etc.
  - streaming Internet

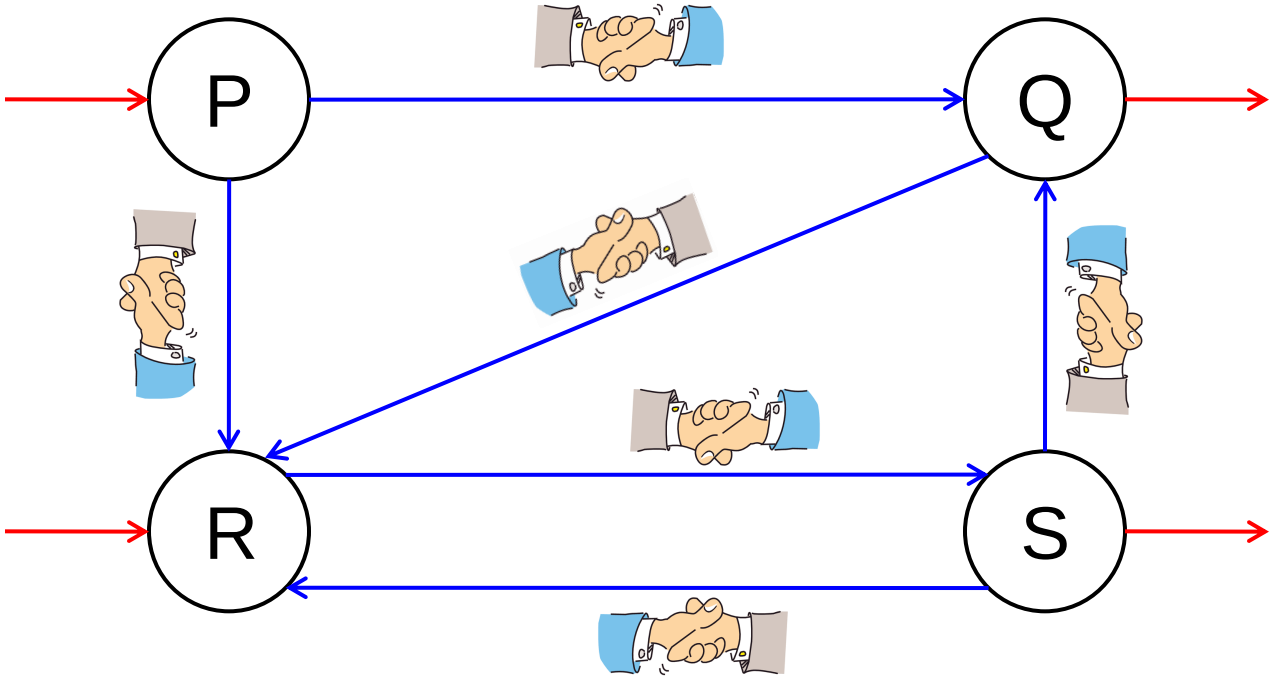
Attention : peu résistant aux changements !
(test entrée vide, mélanges de canaux, etc.)

# *CSP : rendez-vous = information partagée*




- P and Q  sont asynchrone, mais communiquent de façon localement synchrone
- A ce moment, chacun sait ce que sait l'autre
- Les rendez-vous sont ordonnancés de manière non-déterministe

# *CSP = séquentiel + rendez-vous*




t := 10 ;

code de R : P ? x : …
| Q ? y : …
| S ! 10 : …

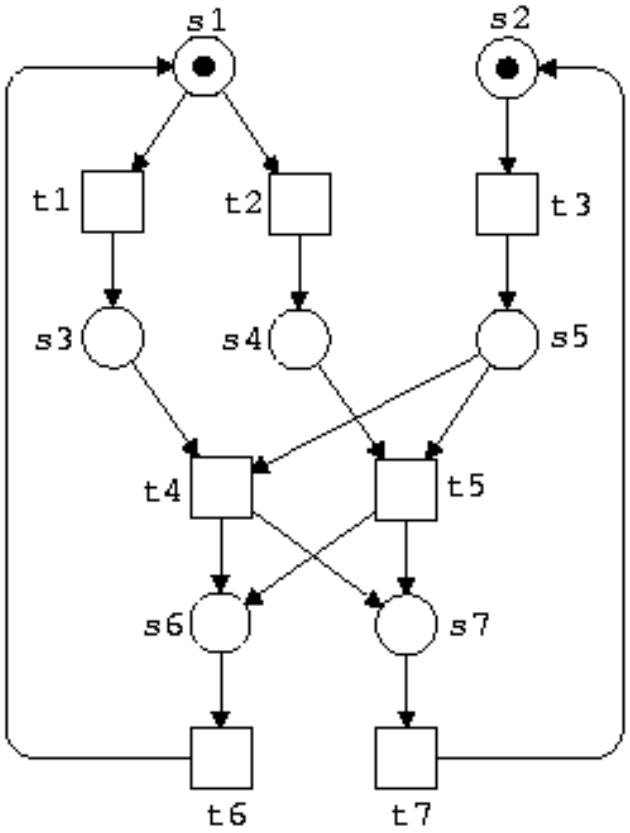
code de S : Q ! (z+1) : …
| R ? t : …
| S ! 43 : …

# *Agenda*

1. Pourquoi le parallélisme
2. De nouvelles questions
3. Extensions de la programmation séquentielle
4. Réseaux graphiques
5. Calculs de processus et modèles chimiques
6. Le pi-calcul et ses variantes

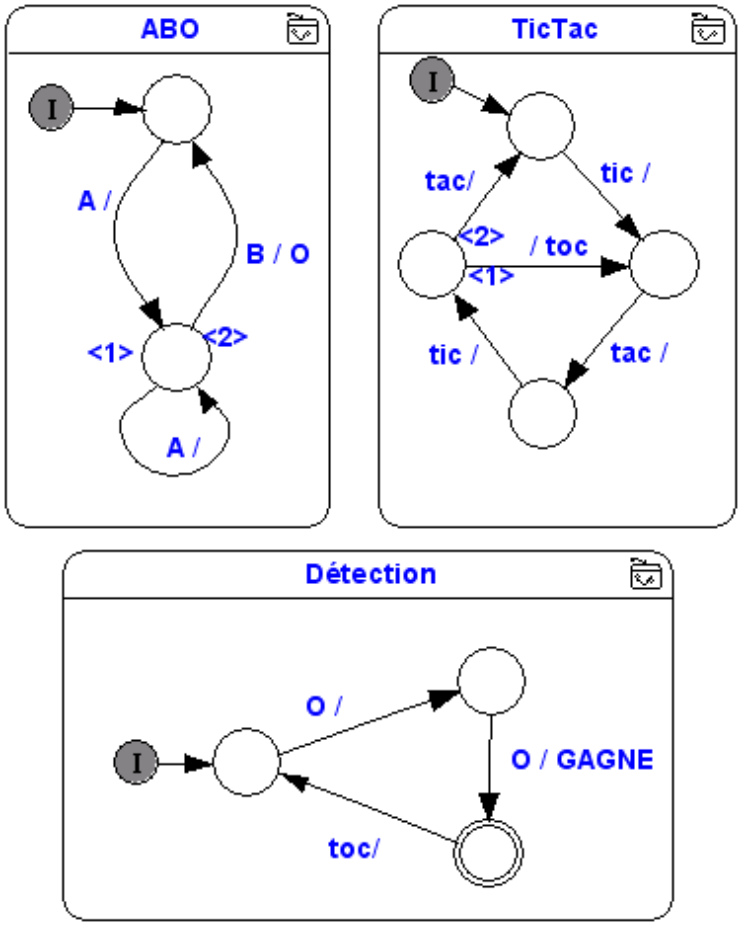
# *Réseaux graphiques*

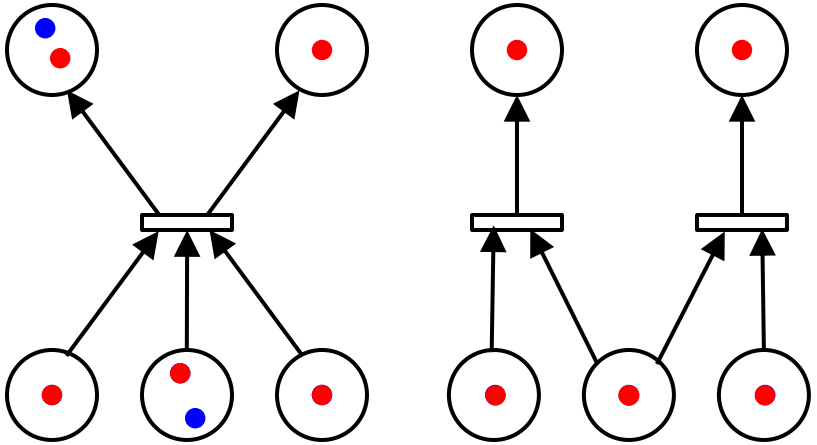
## Réseaux de Petri




Source Univ. Stuttgart

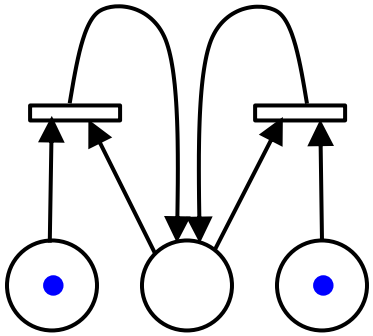
## Réseaux d'automates



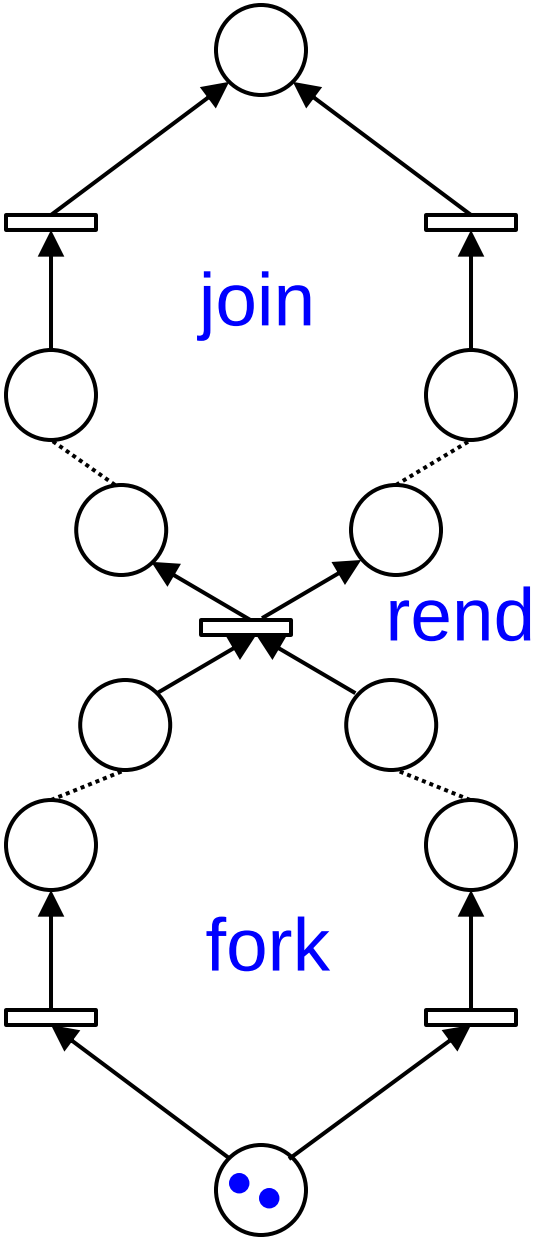


transitions

conflit



interblocage
(deadlock)



# *Agenda*

1. Pourquoi le parallélisme
2. De nouvelles questions
3. Extensions de la programmation séquentielle
4. Réseaux graphiques
5. Calculs de processus et modèles chimiques
6. Le pi-calcul et ses variantes

# *Calculs de processus : CCS (Meije, Lotos,..)*

- Actions $A = \{a,b,...\}$, $A^- = \{a^-,b^-,\ldots\}$, $\tau$ (action invisible)

$$\alpha \in A \cup A^-,\ \ \alpha^- : a \leftrightarrow a^- ,\ \mu \in A \cup A^- \cup \{\tau\}$$

- Identificateurs de processus x, y, z
- Processus p, q, r,… définis par

| | |
|---|---|
| $0$ | inaction |
| $\mu.p$ | action |
| $p \mid q$ | parallélisme |
| $p + q$ | choix non-déterministe |
| $p \setminus a$ | restriction |
| $x$ | identificateur de processus |
| rec $x = p$ | définition récursive |

# *Sémantique Opérationnelle Structurelle (SOS – G. Plotkin)*

$$\mu.p \xrightarrow{\mu} p$$

$$\frac{p \xrightarrow{\mu} p'}{p+q \xrightarrow{\mu} p'} \qquad \frac{q \xrightarrow{\mu} q'}{p+q \xrightarrow{\mu} q'}$$

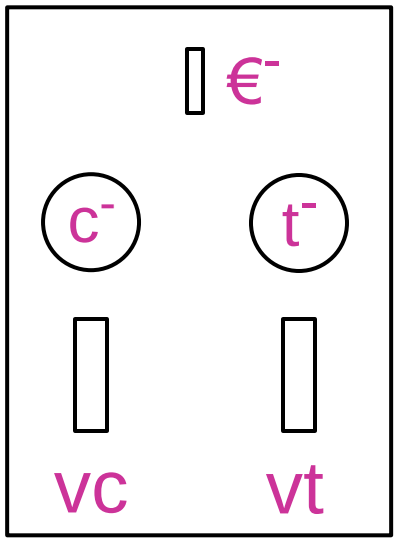
$$\frac{p \xrightarrow{\mu} p'}{p|q \xrightarrow{\mu} p'|q} \qquad \frac{q \xrightarrow{\mu} q'}{p|q \xrightarrow{\mu} p|q'}$$

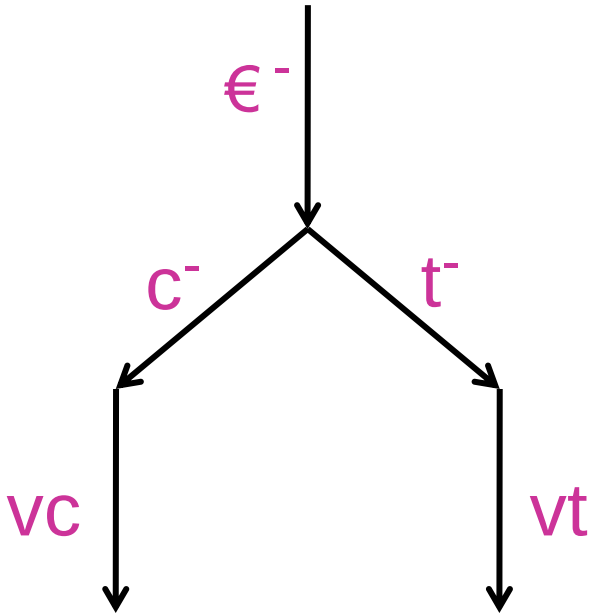
$$\frac{p \xrightarrow{\alpha} p' \quad q \xrightarrow{\alpha^-} q'}{p|q \xrightarrow{\tau} p'|q'}$$

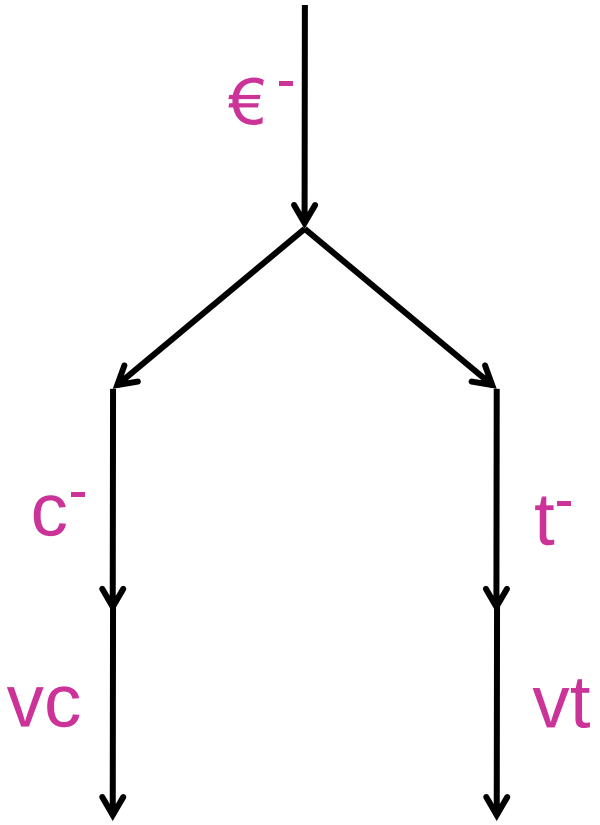
$$\frac{p \xrightarrow{\mu} p' \quad \mu \neq a, a^-}{p \setminus a \xrightarrow{\mu} p' \setminus a}$$

$$\frac{p \xrightarrow{\mu} p'}{\text{rec } x{=}p \xrightarrow{\mu} p' \; [x \leftarrow \text{rec } x{=}p]}$$

# *Deux machines à café / thé*




Laquelle préférez vous ?

# *Exemples*

- Les distributeurs de boissons :

$$\text{rec } dist\_d = €^-.(c^-.vc.dist\_d + t^-.vt.dist\_d)$$

$$dist\_d \xrightarrow{€^-} c^-.vc.dist\_d + t^-.vt.dist\_d$$

$$\text{rec } dist\_n = €^-.c^-.vc.dist\_n + €^-.t^-.vt.dist\_n)$$

$$dist\_n \xrightarrow{€^-} c^-.vc.dist\_n$$

$$dist\_n \xrightarrow{€^-} t^-.vt.dist\_n$$

- Pile standard :

$$\text{rec } PileX = courant.PileX + \tau.0$$

- La pile Wonder ne s'use que si l'on s'en sert :

$$\text{rec } Wonder = courant.(Wonder + \tau.0)$$

# *La bisimulation*

- dist_d et dist_n engendrent le même langage mais ont des comportements interactifs différents
- La bisimulation (forte) compare les possibilités, pas seulement les actions effectuées.

  C'est la plus grande équivalence vérifiant

$$p \approx q \text{ ssi } \begin{cases} p \xrightarrow{\alpha} p' \Rightarrow \exists q'.\ q \xrightarrow{\alpha} q' \text{ avec } q' \approx p' \\ q \xrightarrow{\alpha} q' \Rightarrow \exists p'.\ p \xrightarrow{\alpha} p' \text{ avec } p' \approx q' \end{cases}$$

$$\text{dist\_d} \xrightarrow{€^-} c^-.vc.\text{dist\_d} + t^-.vt.\text{dist\_d} \xrightarrow{t^-} vt.\text{dist\_d}$$

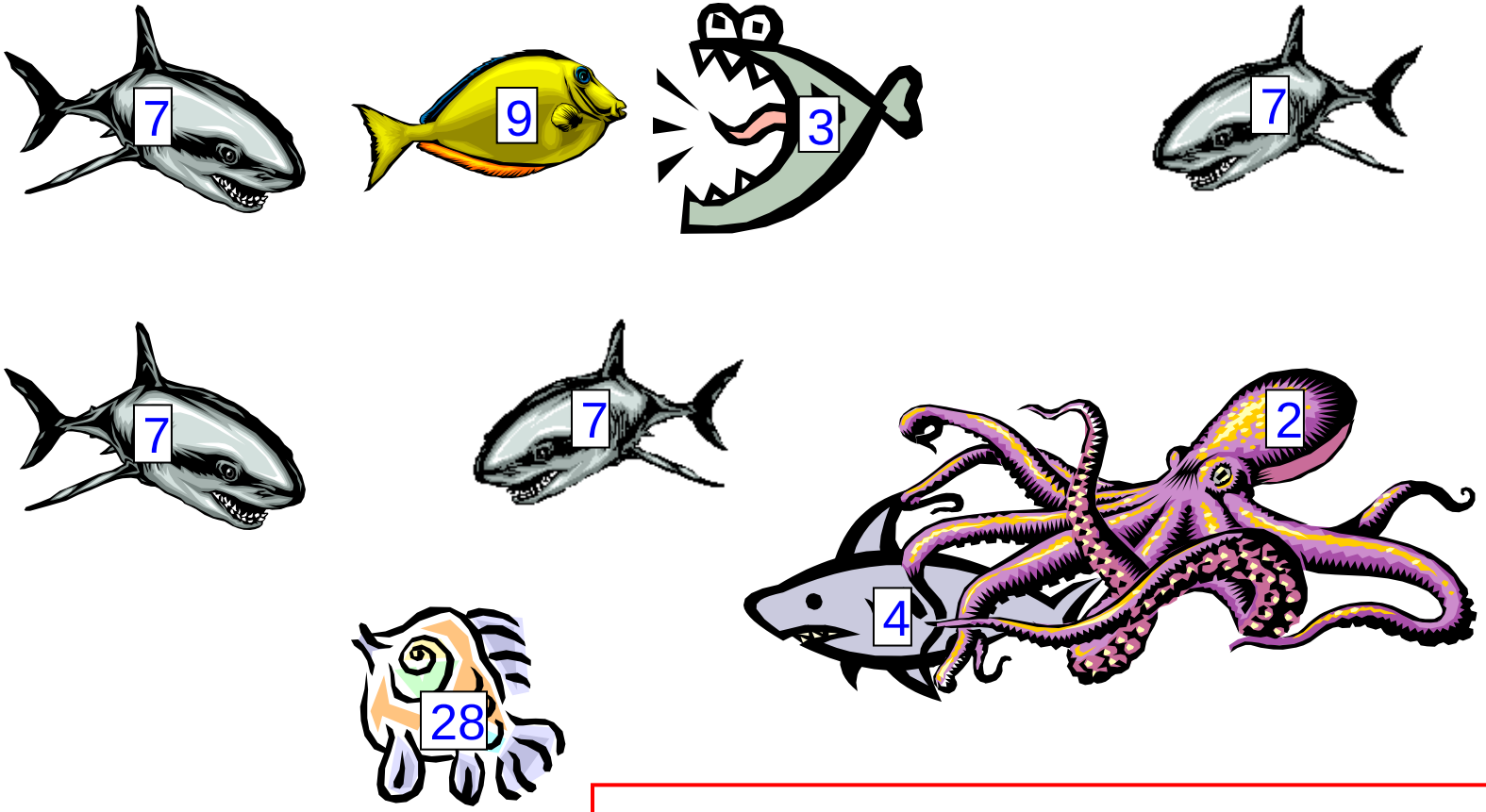
$$\text{dist\_n} \xrightarrow{€^-} c^-.vc.\text{dist\_d} \qquad \not\xrightarrow{t^-}$$

# SOS: simple et précis, mais lourd

$$a.0 \mid (b.a^-.c.0 \mid b^-.0 \mid p)\backslash b \xrightarrow{\tau} a.0 \mid (a^-.c.0 \mid 0 \mid p)\backslash b \xrightarrow{\tau} 0 \mid (c.0 \mid 0 \mid p)\backslash b$$

$$\dfrac{\dfrac{\dfrac{\dfrac{b.a^-.c.0 \xrightarrow{b} a^-.c.0 \qquad b^-.0 \xrightarrow{b^-} 0}{b.a^-.c.0 \mid b^-.0 \xrightarrow{\tau} a^-.c.0 \mid 0}}{b.a^-.c.0 \mid b^-.0 \mid p \xrightarrow{\tau} a^-.c.0 \mid 0 \mid p}}{(b.a^-.c.0 \mid b^-.0 \mid p)\backslash b \xrightarrow{\tau} (a^-.c.0 \mid 0 \mid p)\backslash b}}{\dfrac{\dfrac{a.0 \xrightarrow{a} 0 \qquad \dfrac{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots}{(a^-.c.0 \mid 0 \mid p)\backslash b \xrightarrow{a^-} (c.0 \mid 0 \mid p)\backslash b}}{a.0 \mid (a^-.c.0 \mid 0 \mid p)\backslash b \xrightarrow{\tau} 0 \mid (c.0 \mid 0 \mid p)\backslash b}}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots}}$$

# *Le crible de Darwin : p, kp → p*

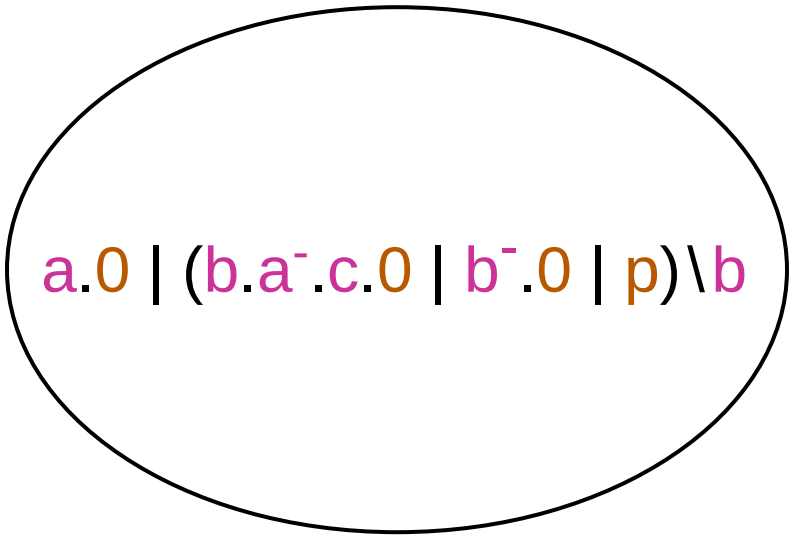



Banâtre - Le Métayer : GAMMA

Berry - Boudol : CHAM

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

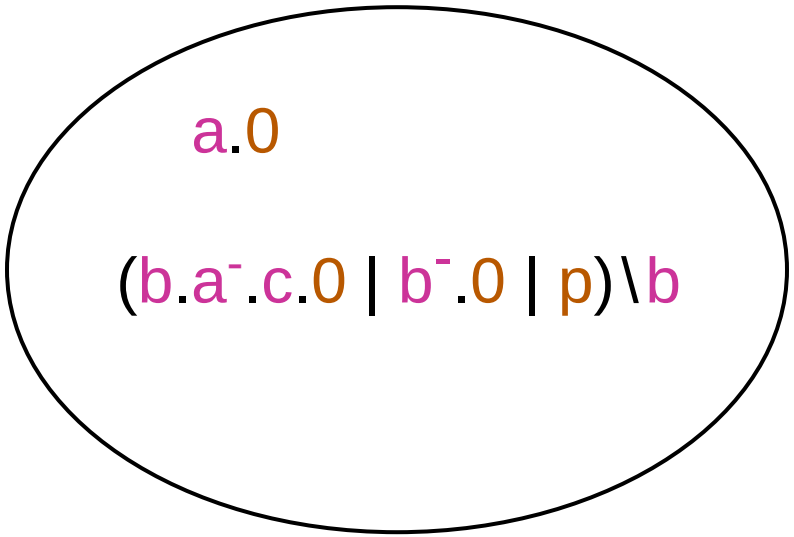



a.0 | (b.a⁻.c.0 | b⁻.0 | p)\b

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution

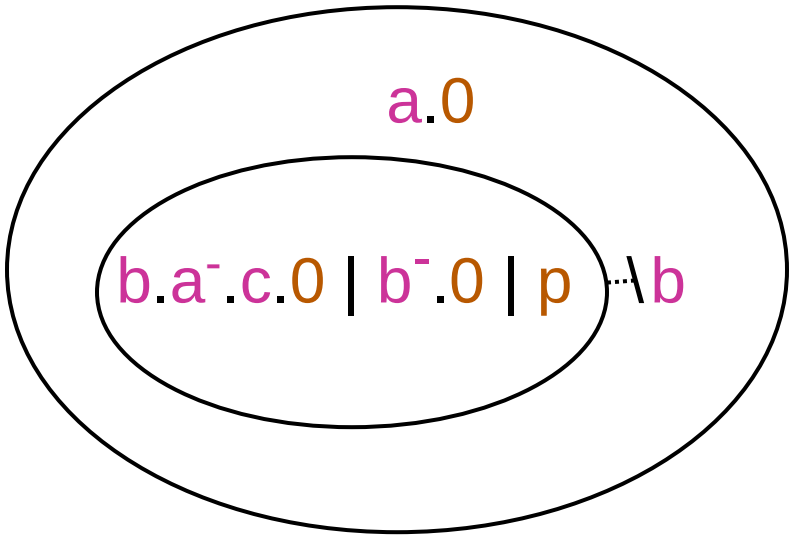



a.0 | (b.$a^-$.c.0 | $b^-$.0 | p)\b

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - $a$, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane

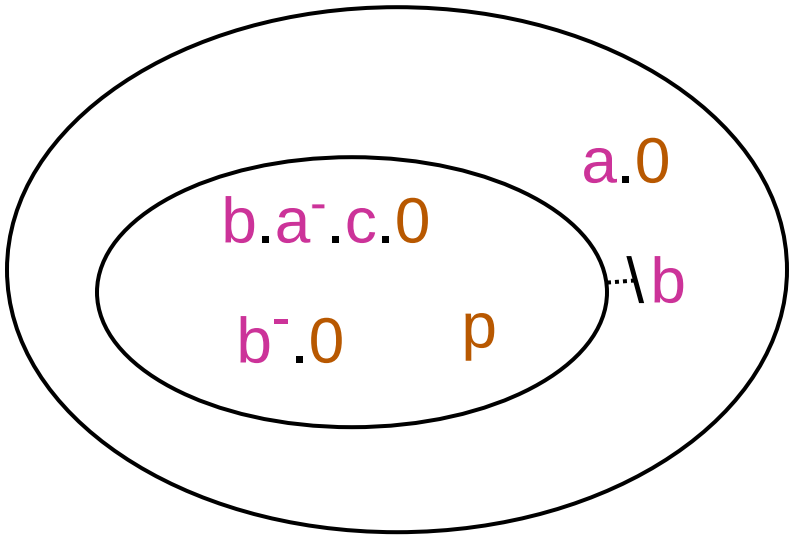



$a.0 \mid (b.a^-.c.0 \mid b^-.0 \mid p) \backslash b$

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution

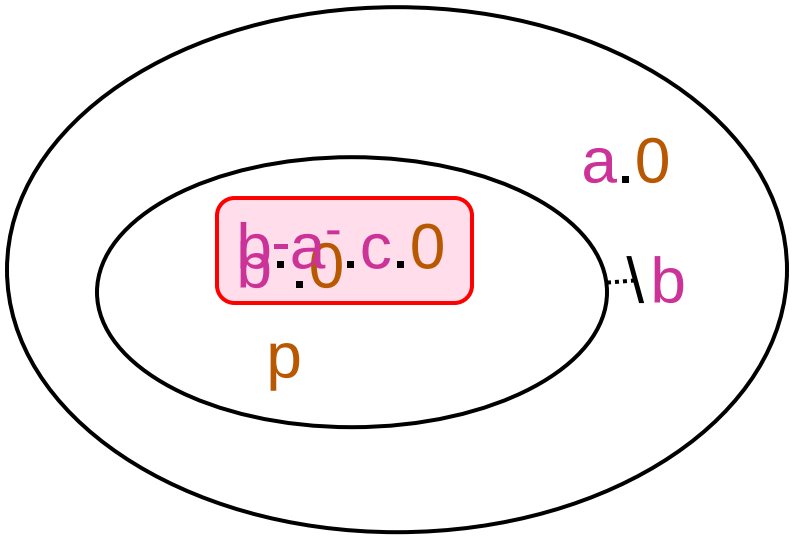



$a.0 \mid (b.a^-.c.0 \mid b^-.0 \mid p)\backslash b$

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre

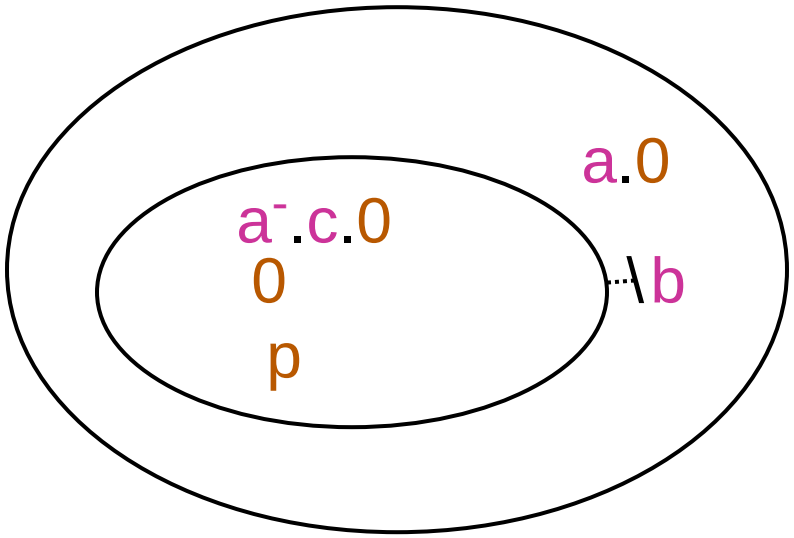



$a.0 \mid (b.a^-.c.0 \mid b^-.0 \mid p) \backslash b$

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre
réaction

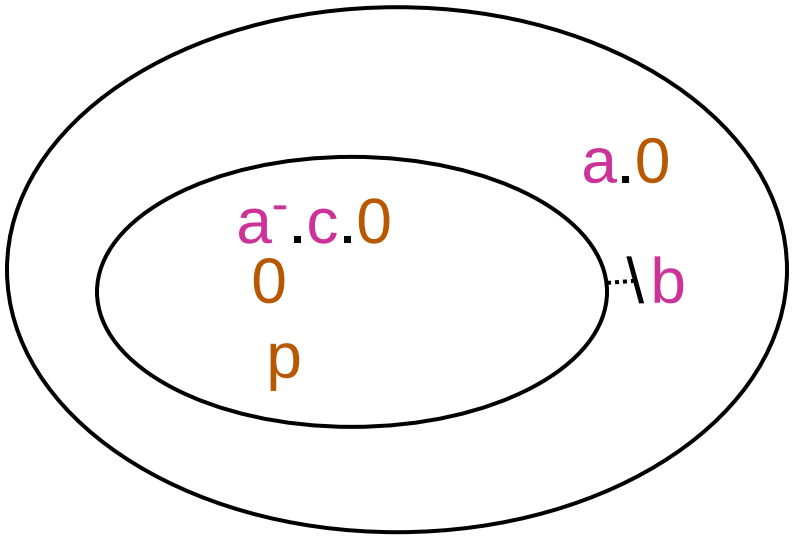



a.0 | (a-.c.0 | 0 | p)\b

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre
réaction
évaporation

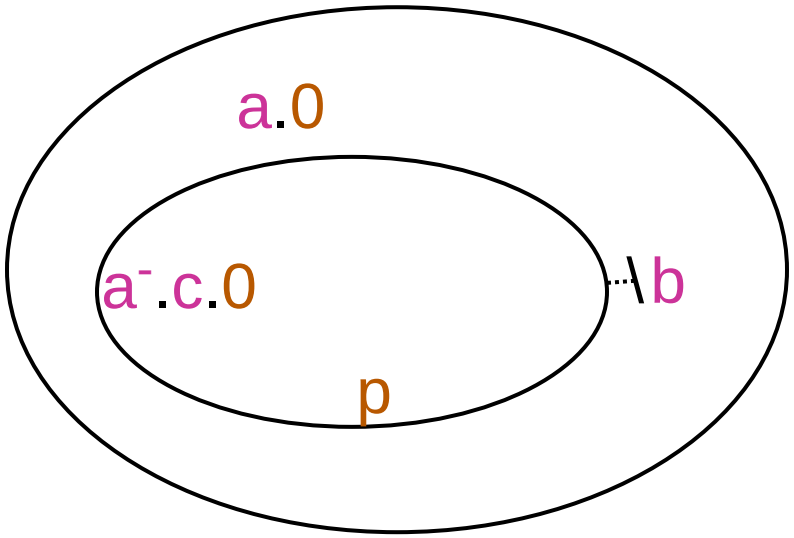



a.0 | (a-.c.0 | p)\b

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre
réaction
évaporation

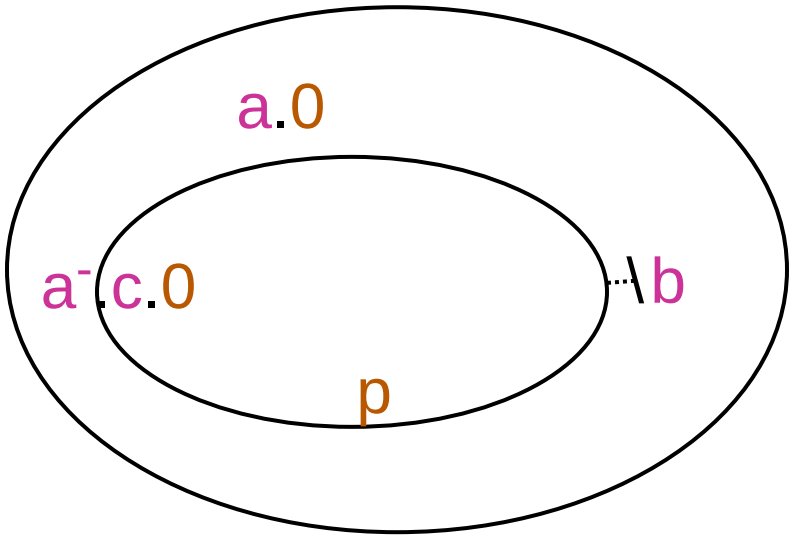



a.0 | (a-.c.0 | p)\b

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre
réaction
évaporation



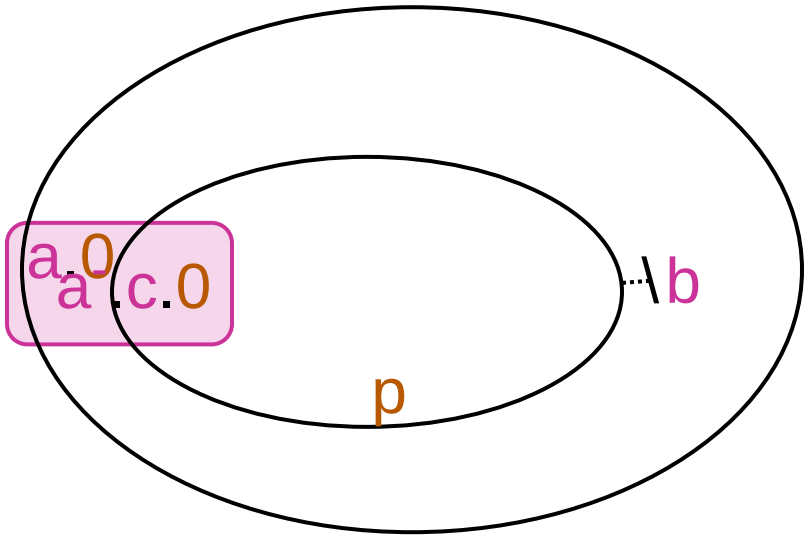

extrusion

a.0 | ($a^-$.c.0 | p)\b

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre
réaction
évaporation



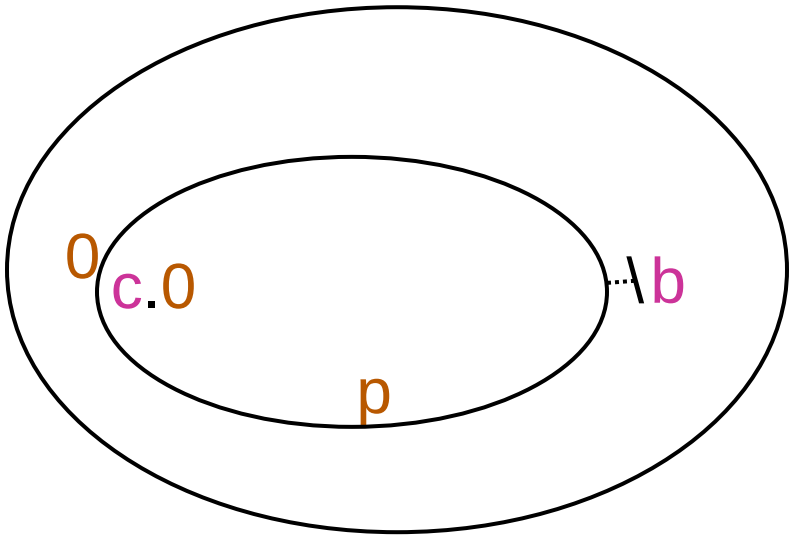

extrusion
rencontre

$a.0 \mid (a^-.c.0 \mid p) \backslash b$

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre
réaction
évaporation



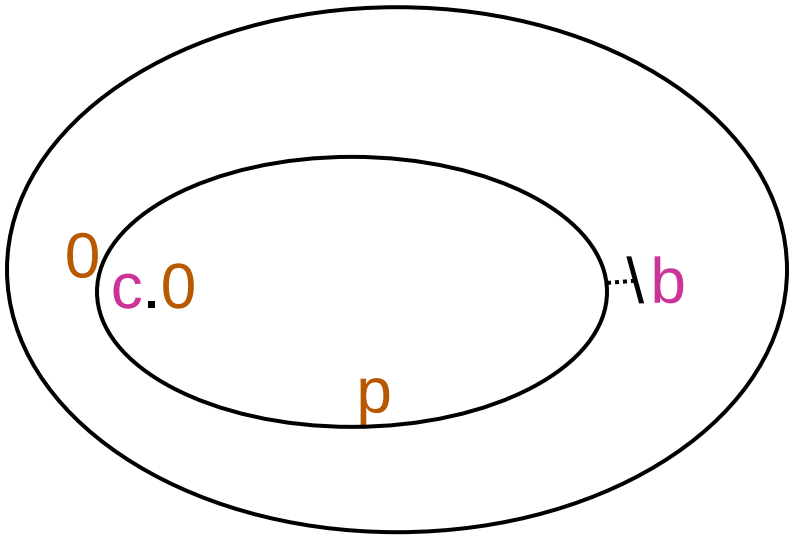

extrusion
rencontre
réaction

0 | (c.0 | p)\b

# *La machine chimique*

- Idée centrales :
  - objets = molécules flottant dans une soupe chimique
  - a, $a^-$ = valences des molécules
  - hiérarchisation => membranes perméables aux valences

dissolution
membrane
dissolution
rencontre
réaction
évaporation




extrusion
rencontre
réaction
évaporation

(c.0 | p)\b

# *Agenda*

1. Pourquoi le parallélisme
2. De nouvelles questions
3. Extensions de la programmation séquentielle
4. Réseaux graphiques
5. Calculs de processus et modèles chimiques
6. Le pi-calcul et ses variantes

# *Le π-calcul (Milner et. al.)*

- Idées centrales :
  - se rapprocher du λ-calcul : valeurs, lieurs, dynamique,...
  - mais ne travailler qu'avec des noms, pas avec des termes
  - permettre de coder à la fois le λ-calcul et le parallélisme
  - permettre la migration géographique des données et calculs
  - permettre la sécurisation des communications

- La version originale
  - CCS (rendez-vous et somme) + un peu de λ-calcul
  - transitions étiquetées, règles SOS, bisimulations

  ⇒ pas mal de points de confusion...

# *Le π-calcul de Milner*

bob =

. ad<alice, bob@cdf.fr>

. bob@cdf.fr(a)

. a<bonjour, bob@cdf.fr>

. bob@cdf.fr(m)

annuaire = !

ad(n, r)

. ( (n = alice).r<alice@inria.fr>

+ (n = bob).r<bob@cdf.fr>

)

alice = !

alice@inria.fr(m, x)

. x<“vous habitez chez vos parents?”>

# *Création et exfiltration de nom*

```
bob = new r1, r2
   . ad<alice, r1>
   . r1(a)
   . a<bonjour, r2>
   . r2(m)
```

```
annuaire = !
  ad(n, r)
    . (  (n = alice).r<alice@inria.fr>
       + (n = bob).r<bob@cdf.fr>
      )
```

```
alice = !
 alice@inria.fr(m, x)
 . x<“vous habitez chez vos parents?”>
```

# *Le π-calcul asynchrone (Boudol, Honda)*

- Inconvénients du π-calcul initial :
  - rendez-vous peu naturel et antagoniste à distribution
  - choix non-déterministe '+' ambigu
    non-déterminisme externe ou interne?
  - mélange input / output au même niveau
    ⇒ synchronisation globale
  - utilisation arbitraire des noms ⇒ trous de sécurité

- Evolution : vrai asynchrone + chimie ⇒ distribution
  - π-calculs asynchrones (Boudol / Honda / Sangiorgi) :
    lâcher les messages dans la soupe chimique, les laisser migrer tous seuls vers leurs récepteurs
  - join-calcul (Fournet Gonthier Lévy)

# Le $L\pi$-calcul

| | |
|---|---|
| 0 | inaction |
| a<b> | message |
| a(x).p | récepteur |
| !a(x).p | serveur |
| p \| q | parallèle |
| (νa) p | générateur de nom |

x uniquement en position émettrice x<.> dans p

$p \mid q \equiv q \mid p$ $\quad (p \mid q) \mid r \equiv (p \mid q) \mid r$ $\quad p \mid 0 \equiv p$

$(\nu a)\, 0 \equiv 0$ $\quad (\nu a)(\nu b)\, p \equiv (\nu b)(\nu a)\, p$

$((\nu a)\, p) \mid q \equiv (\nu a)\, (p \mid q)$ si a non libre dans q

$a(x).p \mid a\langle b\rangle \rightarrow p[b/x]$

$!a(x).p \mid a\langle b\rangle \rightarrow p[b/x] \mid !a(x).p$

# *Conclusion*

- Le calcul parallèle asynchrone est beaucoup plus riche et complexe que le calcul séquentiel
- La mémoire partagée revient à la mode mais reste fort complexe
- Il existe de nombreux autres modèles partiels utiles
- Les modèles les plus ambitieux sont les $\pi$-calculs utilisés en sécurité, modélisation biologique, etc.

Mais la question du sens même de la modélisation reste posée !

# *Références*

- *The Art of Multiprocessor Programming*
  Maurice Herlihy et Nir Shavit
  Morgan Kaufmann, 2008

- *Synchronisation dans les systèmes répartis*
  Michel Raynal
  Eyrolles, 1992

- *Kahn Networks at the Dawn of Functional Programming*
  David McQueen
  Dans *From Semantics to Computer Science*,
  Morgan-Kaufmann, 2009

# *Références*

- *A Calculus of Communicating Systems*
  Robin Milner
  Springer-Verlag, 1980

- *The Chemical Abstract Machine*
  Gérard Berry et Gérard Boudol
  Theoretical Computer Science, vol. 96 (1992) 217-248.

- *The Pi-Calculus: A Theory of Mobile Processes*
  Davide Sangiorgi,David Walker
  Cambridge University Press, 2003