# Les Réseaux Synchronisés

Nicolas Guelfi



# Introduction

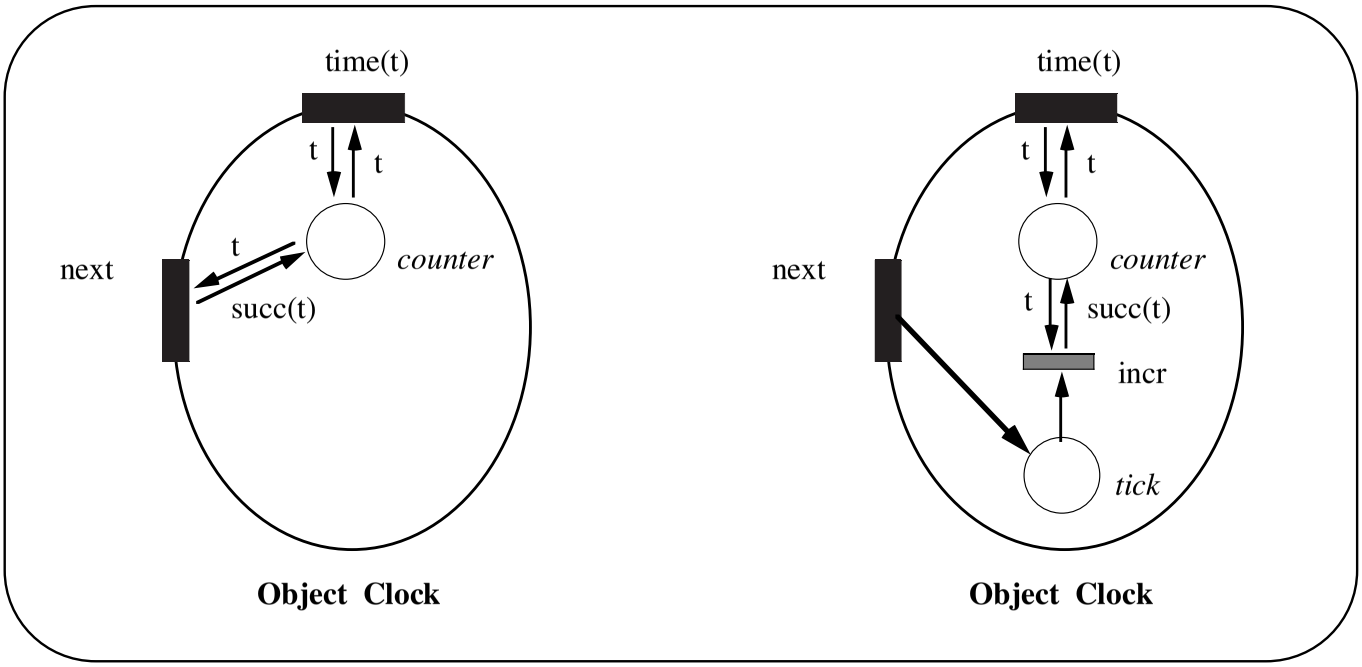
- **Comment résoudre les problèmes de :**
  - **synchronisation libre**
  - **stabilisation**
  - 
  - **CO-OPN en général • • •**

# PLAN

- stabilisation
  - » **exemple connu**
  - » **définition**
  - » **problème**
  - » **historique**
- sémantique des RS
  - » **exemples intuitifs**
  - » **règles sémantiques**
  - » **réduction à un problème de théorie des graphes**



# La (les) stabilisation(s) : exemple



# La (les) stabilisation(s)

- Définition : un module est stable lorsque aucun de ses évènements internes est exécutable
- Remarques :
  - » **abstraction des comportements**
  - » **pré-condition supplémentaire à l'exécution des méthodes du module**
  - » **élément global d'implémentation des méthodes**



# Le problème de la sémantique synchrone stable (PSSS)

- pour permettre une synchronisation, il faut être stable
- pour être stable, il faut parfois se synchroniser

# CO-OPN ' 90

- les objets
  - » **méthodes externes ou internes**
- la composition
  - » **synchronisation devant induire une hiérarchie stricte sur les objets**
- la stabilisation
  - » **incomplète et semi-formelle**
  - » **dépendante de la hiérarchie**



# CO-OPN ' 94 : HAN

- les objets
  - » **méthodes externes ou internes**
- la composition
  - » **opérateurs syntaxiques définissant la hiérarchie sur les objets**
- la stabilisation
  - » **incomplète et formelle**
  - » **dépendante de la hiérarchie**

# CO-OPN ' 95 : RS

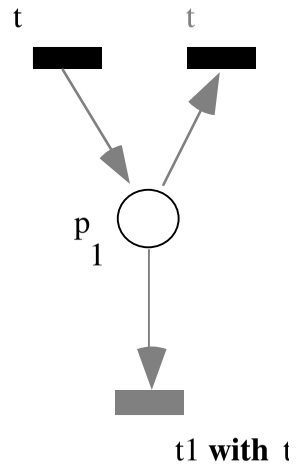
- les objets
  - » **méthodes externes ou internes**
- la composition
  - » **union de module**
  - » **graphe libre de synchronisations**
- la stabilisation
  - » **complète et formelle**
  - » **indépendante du type de graphe**



# CO-OPN ' 96 : CO-OPN/2

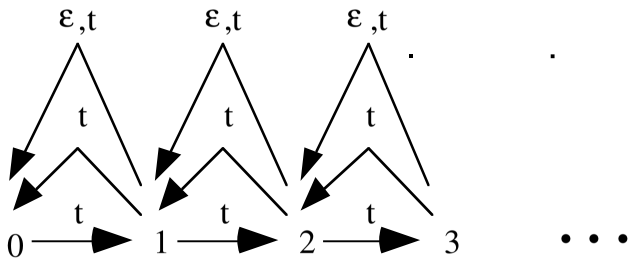
- les objets
  - » **méthodes externes ou internes (1 + 2 ?)**
- la composition
  - » **union de module**
  - » **graphe libre de synchronisations**
- la stabilisation
  - » **complète et formelle**
  - » **indépendante du type de graphe**
  - »
- => modèle d'exécution fondé sur les RS
  - » **quelle stabilisation pour quelles méthodes internes?**

# PSSS exemple : 1 (a)



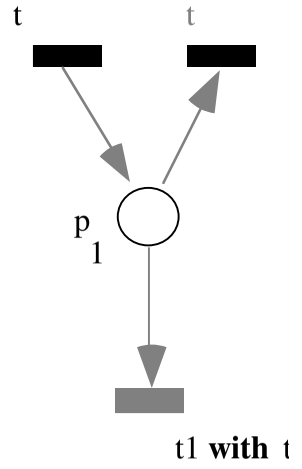

Mod e

(Elementary model)

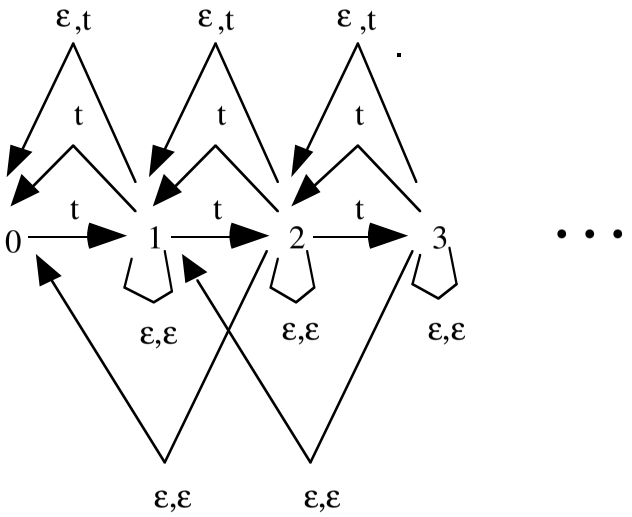




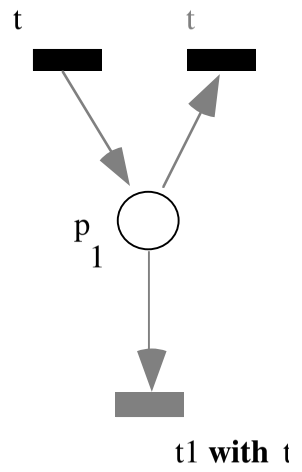

# PSSS exemple : 1 (b)

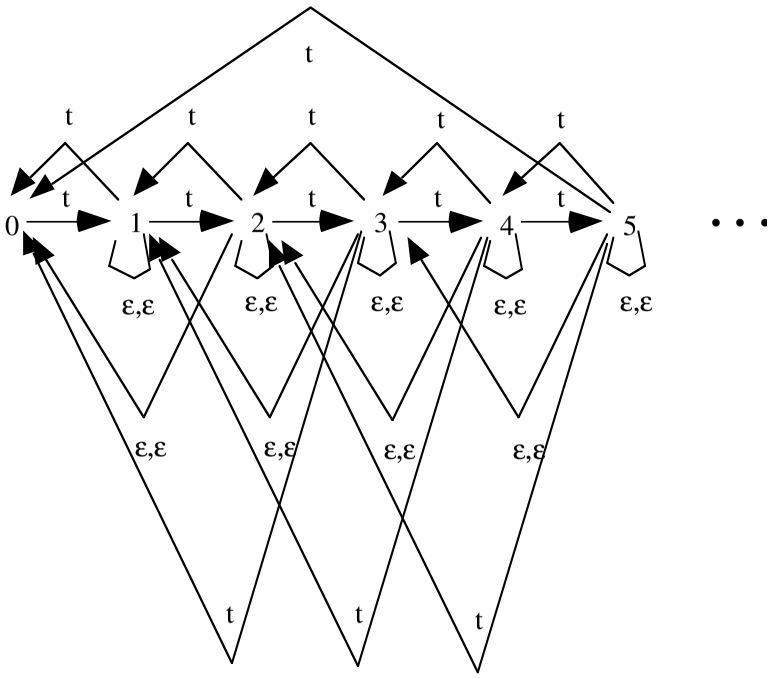



Sync 1



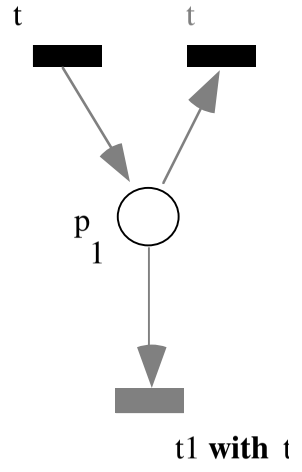
# PSSS exemple : 1 (c)

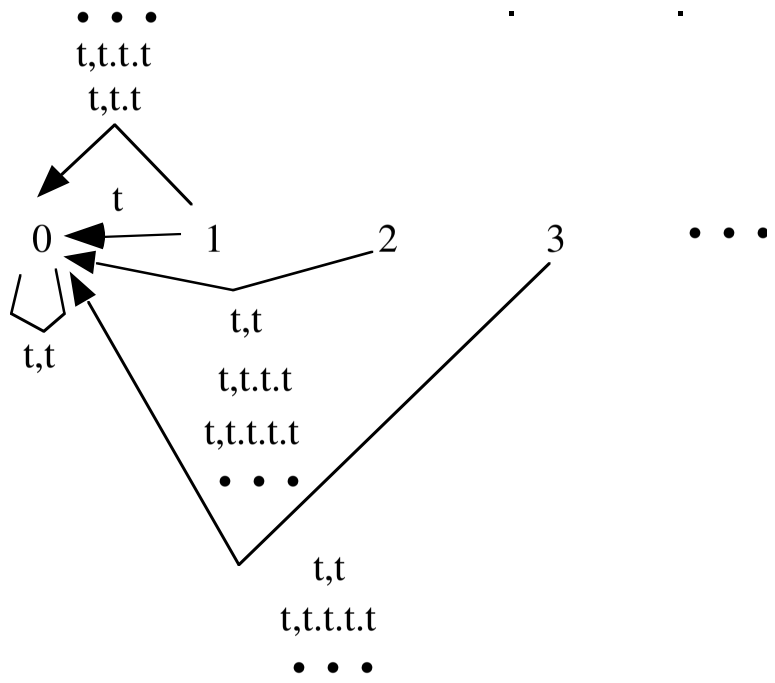


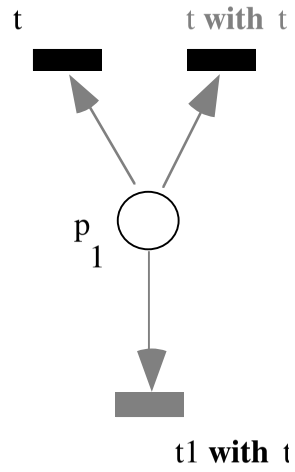


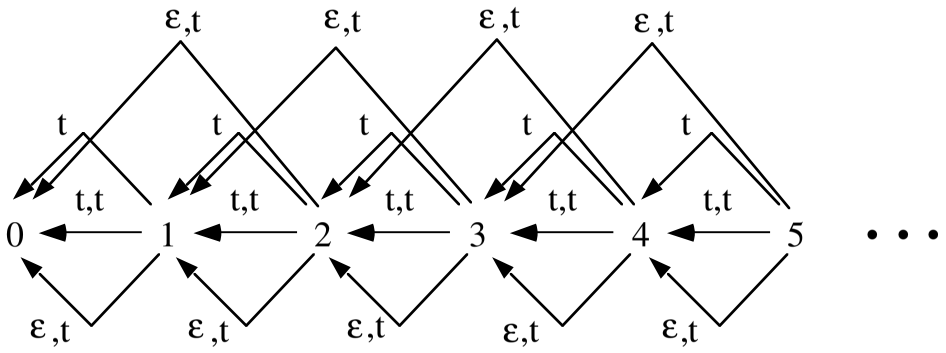

# PSSS exemple (RAH) : 1 (d)



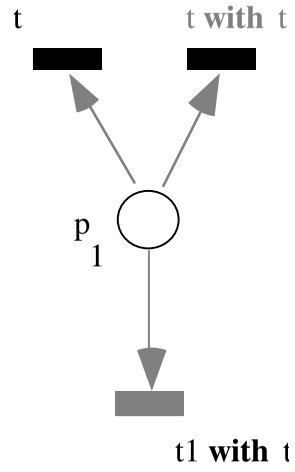
# PSSS exemple : 2 (a)

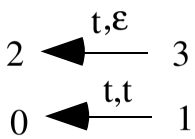
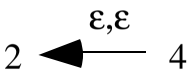




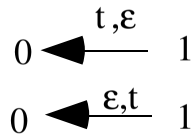
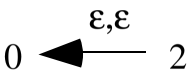

# PSSS exemple : 2 (b)




(a) synchro de  donne 





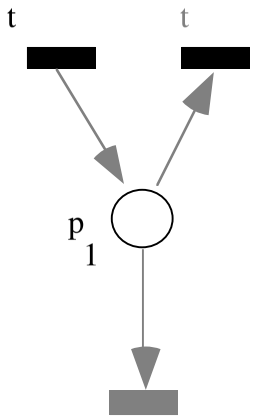
(b) synchro de  donne 



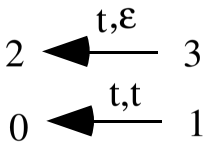


=> **2 est un état qui est devenu instable après avoir utilisé des transitions qui y menaient**

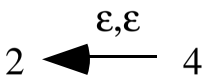
# PSSS exemple : 2 (c)



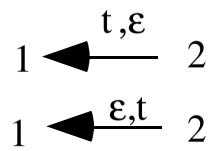

(a) synchro de



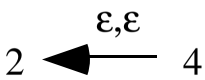

donne




(b) synchro de




donne




=> **4 est un état faussement instable car on a utilisé un état instable pour le rendre instable**



# Cloture Comportementale (1)

$$ID \ : \ \left[ \frac{mr_1 \xrightarrow{<proc_1, proc'_1>} mr'_1}{mr_1 \xrightarrow{<proc_1, proc'_1>, \emptyset} mr'_1} \right]$$

$$(proc_1 \neq \varepsilon) \ \wedge \ (proc_2 \neq \varepsilon)$$

$$SEQ \ : \ \left[ \frac{mr_1 \xrightarrow{<proc_1, \varepsilon>, M_1} mr'_1 \ \wedge \ mr'_1 \xrightarrow{<proc_2, \varepsilon>, M_2} mr'_2}{mr_1 \xrightarrow{<seq(proc_1, proc_2), \varepsilon>, M_1 \cup M_2 \cup \{mr_1, mr'_1, mr'_2\}} mr'_2} \right]$$

# Cloture Comportementale (2)

$$(proc_1 \neq \varepsilon)$$

$$STAB \ : \ \left[ \frac{mr_1 \xrightarrow{<proc_1, \varepsilon>, M_1} mr_1' \ \wedge \ mr_1' \xrightarrow{<\varepsilon, \varepsilon>, M_2} mr_2'}{mr_1 \xrightarrow{<proc_1, \varepsilon>, M_1 \cup M_2} mr_2'} \right]$$



# Cloture Comportementale (3)

$$(proc_1 \neq \varepsilon) \ \wedge \ (proc_2 \neq \varepsilon)$$

$$SIM \ : \ \left[ \frac{mr_1 \xrightarrow{<proc_1, \varepsilon>, M_1} mr_1' \ \wedge \ mr_2 \xrightarrow{<proc_2, \varepsilon>, M_2} mr_2'}{mr_1 + mr_2 \xrightarrow{<sim(proc_1, proc_2), \varepsilon>, M_1 \cup M_2 \cup \{mr_1, mr_2, mr_1', mr_2'\}} mr_1' + mr_2'} \right]$$

# Synchronisation

$$(proc_1 \neq \varepsilon) \wedge (proc_2 \neq \varepsilon) \wedge$$

$$(\forall proc'_2 \in AltEq(proc'_1)) \ / \ proc'_2 \equiv_p proc_2$$

$$SYNC : \left[ \frac{mr_1 \xrightarrow{<proc_1, proc'_1>, M_1} mr'_1 \ \wedge \ mr_2 \xrightarrow{<proc_2, \varepsilon>, M_2} mr'_2}{mr_1 + mr_2 \xrightarrow{<proc_1, \varepsilon>, M_1 \cup M_2 \cup \{mr_1, mr_2, mr'_1, mr'_2\}} mr'_1 + mr'_2} \right]$$



# Solution actuelle

$$Sem\ (st) = \left\{ \begin{array}{l} mr_1 \xrightarrow{<proc_1, \varepsilon>} mr'_1 \ / \\ \\ i)\ il\ existe\ mr_1 \xrightarrow{<proc_1, \varepsilon>, M_1} mr'_1\ dans\ st\ avec\ proc_1 \neq \varepsilon \\ \\ ii)\ tous\ les\ marquages\ mr\ de\ M_1\ sont\ stables \\ (i.e.\ il\ n'y\ a\ pas\ de\ transition\ mr \xrightarrow{<\varepsilon, \varepsilon>, M} mr'\ dans\ st\ tel\ que \\ les\ marquages\ de\ M\ soient\ stables\ !) \end{array} \right\}$$

attention : tout marquage initial doit être stable !

# Conclusion

- solution satisfaisante ?
- Futur
  - » **Théorique**
    - – étude de la modularité de la sémantique
  - » **Pratique**
    - – trouver un algorithme pour le parcours de graphe
    - – Analyser sa complexité