

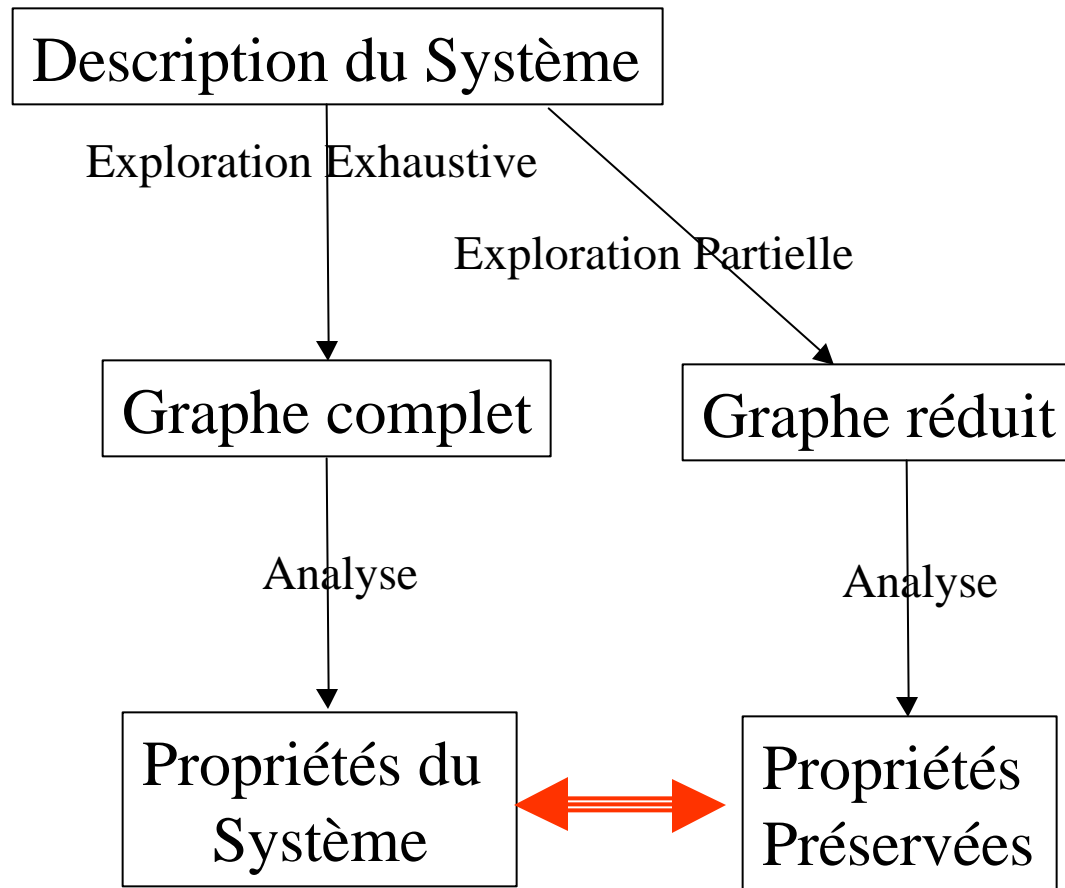
Graphe de Pas Persistant



FAC 2002

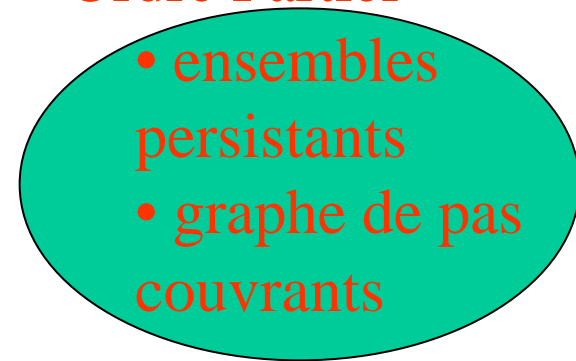
RIBET Pierre-Olivier
VERNADAT François
BERTHOMIEU Bernard

Explosion combinatoire



Méthodes:

- Abstraction
- Symétrie
- **Ordre Partiel**



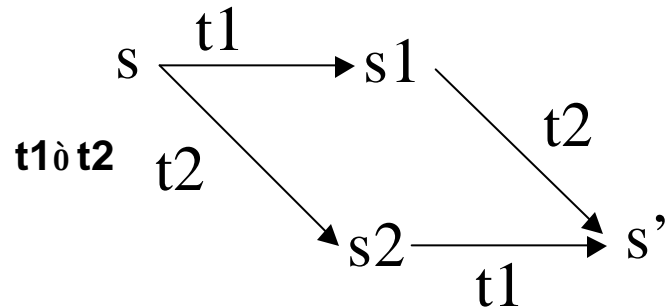
Graphe de pas persistant

Cadre:Préservation des blocages

Analyse=Logique temporelle, bisimulation

Approches ordres partiel 1/2

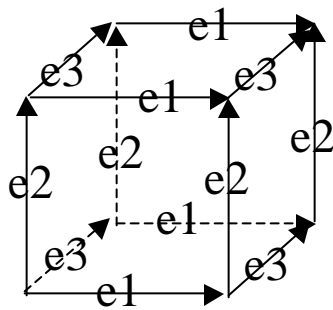
Deux transitions indépendantes confluent



ò peut être obtenue structurellement
(relation complémentaire: #)

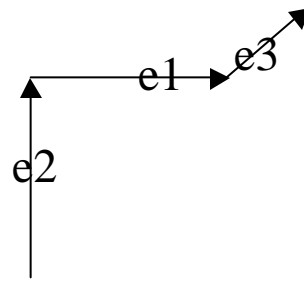
⊢ il peut suffire de visiter un chemin par trace

Exhaustive



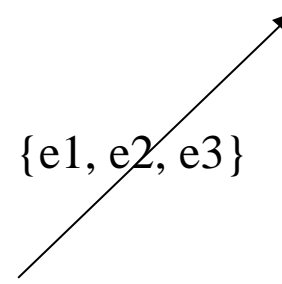
- 2^n états
- $n \cdot 2^{n-1}$ arcs

1 chemin par trace



- $(n+1)$ états
- n arcs

Pas Couvrant (LAAS)

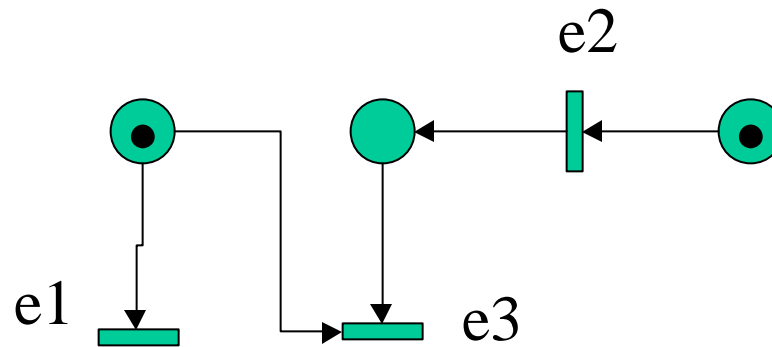
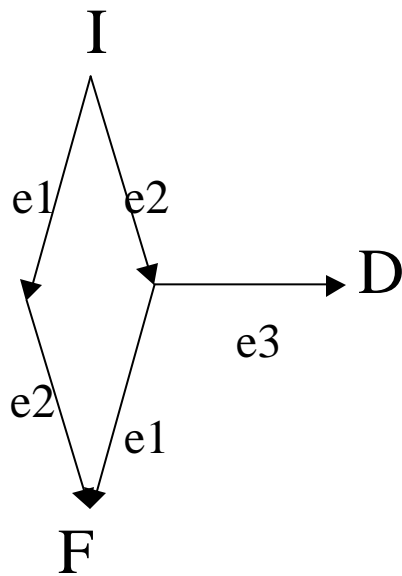


- 2 états
- 1 arc

Approche ordre partiel 2/2

⇒ La confluence ne suffit pas à la réduction.

Cas de Confusion



Analyse exhaustive de systèmes

Semi-algorithme d'exploration:

```
Queue ← s0
H ← {s0}
G ← ∅
While NotEmpty(Queue) do
    s ← dequeue(Queue)
    if Enabled(s) = ∅ then
        print « Blocage »
    else
        Explore_exhaustive(Enabled(s))
```

Explore_exhaustive(T):

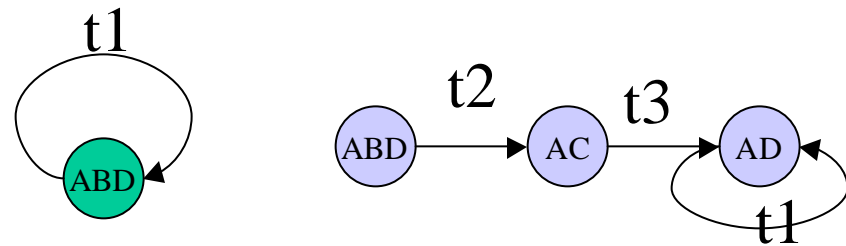
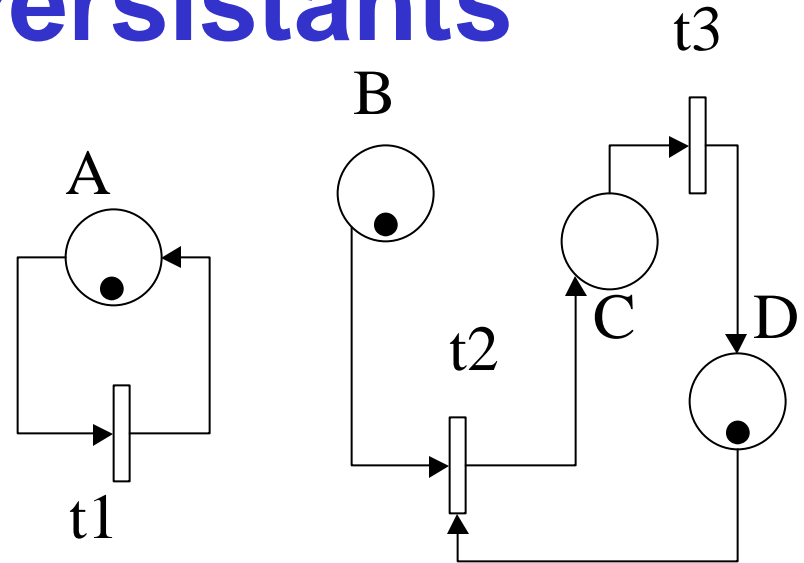
```
foreach t in T do
    s' ← fire(s,t)
    G ← G U {<s,t,s'>}
    if s' ∉ H then
        H ← H U {s'}
        enqueue(Queue,s')
```

Redéfinition de Explore_exhaustive pour les explorations partielles

Ensembles Persistants

`explore_persistent(T):`
 $P \leftarrow A(T)$
`Explore_exhaustive(P)`

Intuitivement: « Ensemble qui est indépendant de toutes les autres transitions. »



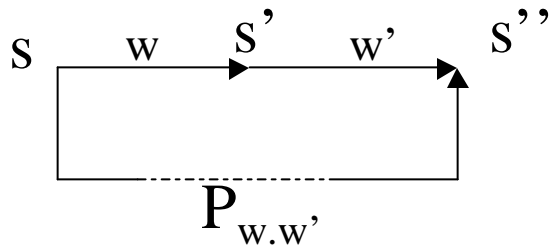
- Enabled(s) est persistant
- $\{t \mid t \text{ [#] } t_0\}$ est persistant

- Choix heuristique
- Non déterministe
- Ignoring Problem

Graphe de Pas Couvrants (GPC)

Intuitivement, un GPC c'est:

- Tout pas correspond à un ensemble de séquences de transitions
- Toute séquence de transitions du graphe complet est préfixe d'une séquence de pas possédant la même trace



Explore_GPC(T):

$T_u \leftarrow TU(T)$

$T_m \leftarrow TM(T)$

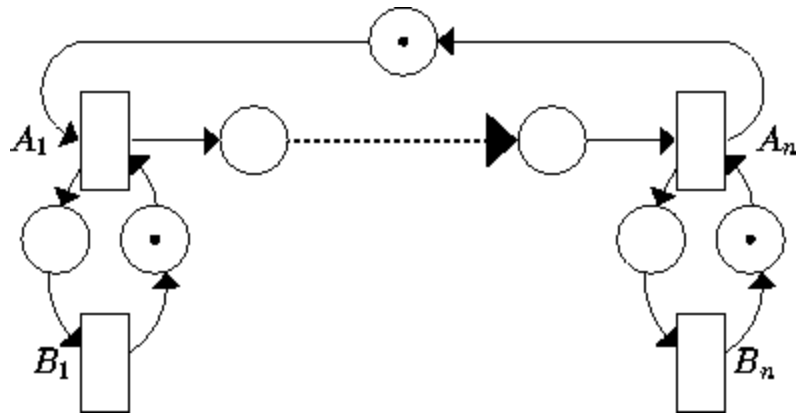
Explore_exhaustive(T_u)

$\Pi \leftarrow \Pi(T_m)$

Explore_par_pas(Π)

- Déterministe
- Pas d'Ignoring problem
- Préservation des blocages, mais aussi des «traces» maximales

Scheduler de Milner GPC



- Graphe exhaustif :

- cas général :

$n \cdot 2^n$ états

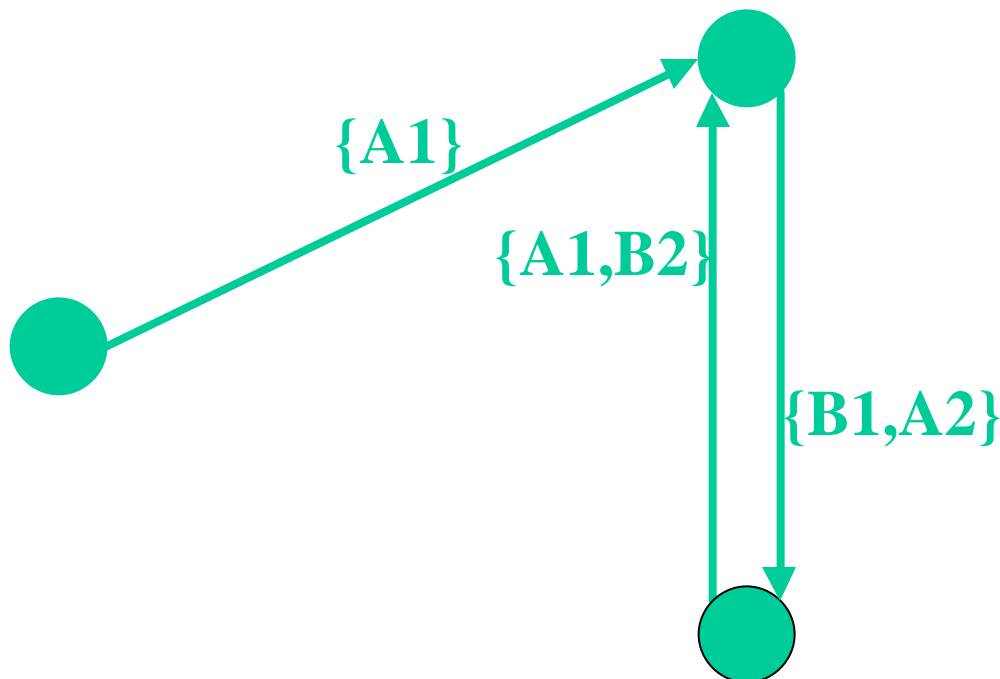
- $n=2$: 8 états

- Graphe de pas couvrants :

- cas général :

$n + 1$ états

- $n=2$: 3 états



Graphe de Pas

Combiner les 2 ?

Modèle	Exhaustif	GP	GPC
Scheduler 300	$2^n * n = 6.10^{92}$	1 394	301
Philosophe 8	103 681	233	31 231
Naimi-Trehel 5	202 500	40 006	52 681

2 idées:

⇒ Combiner ces deux méthodes -> méthode plus performante

⇒ Simplifier le GPC pour ne préserver que les blocages

Graphe de Pas Persistants (GPP)

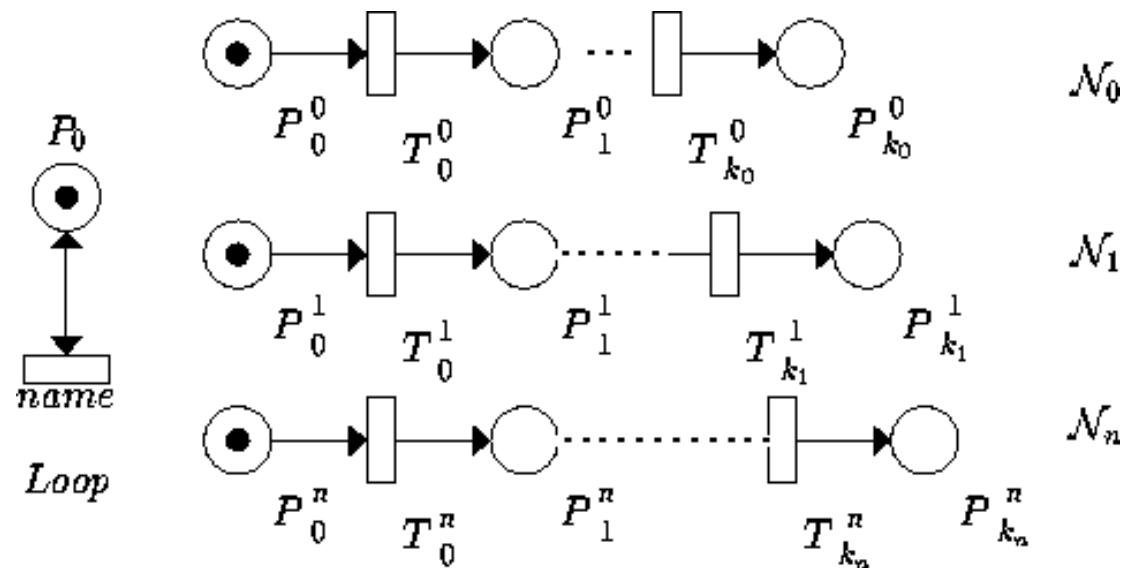
Idée: Les pas sont construits sur un sous-ensemble de Enabled

```
Explore_GPP(T):  
  Tu ← TU(T)  
  Tm ← TM(T)  
  Si Tm = ∅ alors  
    Explore_exhaustive(Tu)  
  Sinon  
    P ← A(T) ∩ Tm   Choix de A  
    Π ← Π(P)       Choix de Π  
    Explore_par_pas(Π)
```

⇒ Différentes instances (plus ou moins déterministes)

- GPP est meilleur que GPC
- GP est une instance de GPP ⇒ GPP aussi bon que GP

Instance de GPP meilleure que tout GP ?



- Exploration PG (Heuristique min)
 - Commence par Loop \rightarrow 1 état \ 1 arc
 - Termine par Loop $\rightarrow \sum_{i=0}^{i=n} k_i + 1$ états
- Exploration GPP (Heuristique min) Déterministe ici
 - Max $\{k_i + 1 : i \in [1, n]\}$ états

Résultats

Modèle	Exhaustif	GP	GPC	GPP
Scheduler 300	$2^n * n = 6.10^{92}$	1 394	301	301
Philosophe 8	103 681	233	31 231	227
Naimi-Trehel 5	202 500	40 006	52 681	40 001

- En pratique, GPP mieux que GP et GPC
 - TINA (Time Petri Nets Analyzer)
 - exploration et analyse pour RdP et RdP temporels
 - téléchargeable
- <http://www.laas.fr/tina>*
- GP, GPC, GPP

Bilan/Perspectives

- Différents GPC pour différentes classes de propriétés
 - Blocages + Traces
 - Bisimulation et Equivalence de Refus
 - **Blocages**
- Comparaison des méthodes d'Ordre Partiel
Trace Automaton obtenu par :
 - GPC
 - GP