# Diagnostic de systèmes à événements discrets
# Approches décentralisées / distribuées

Marie-Odile Cordier / IRISA – Université Rennes1
Équipe DREAM

# *Plan*

Systèmes à événements discrets (SED)

- Modélisation des SED
- Diagnostic de SED :
  - Approche diagnostiqueur (Sampath-Sengupta-Lafortune-…95)
- Diagnosabilité
  - Approche diagnostiqueur (Sampath-Sengupta-Lafortune-…95)

Approches décentralisées

- Différentes approches
  - Avec superviseur ou non
  - Modèle global ou local
  - Quand synchroniser ?
- Trois approches :
  - Debouk, Lafortune et al
  - Fabre et al.
  - Pencolé et al.

# *Modèles de diagnostic / différentes approches IA*

- Modèles prédictifs : on modélise le comportement statique / dynamique du système
  - Approches dites « consistency-based » : comparaison observations et prédictions
    - Modèles / systèmes continus : Modèles qualitatifs (ou semi-quantitatifs)
    - Modèles / systèmes discrets
    - Modèles hybrides
- Modèles explicatifs : on modélise les liens comportement-symptômes (conn. pannes)
  - par exemple graphes causaux temporels
  - Approche abductive
- Modèles associatifs
  - Systèmes experts + temps (G2, Chronos …)
  - Reconnaissance de chroniques (scénarios)

# *Formalismes pour les SED*

- Un SED peut-être représenté par un langage
  - Langage :
    - alphabet : ensemble des événements *E*
    - mot : séquence d'événements
  - Comportement d'un SED
    - ensemble de mots formés sur *E*
    - représente l'ensemble des comportements possibles du système
    - langage « prefix-closed » Prefix-closed

- Formalismes
  - systèmes de transitions
    - automates, automates temporels, automates communicants
    - réseaux de Petri,
  - algèbres de processus

# *Automates*

- Outil permettant de représenter un langage
- Définition classique : G = (X, E, f, x0, Xm)
  - $X$ : ensemble d'états
  - $E$ : ensemble des événements (alphabet)
  - $f$ : fonction de transition $X \times E \rightarrow 2^X$
  - $x_0$ : état initial
  - $X_m$ : ensemble d'états finals (marqués)

+ Fonction $\Gamma$: $X \rightarrow 2^E$

Fait correspondre à un état les événements « actifs »

# *Opérations sur les modèles*

- Système : ensemble de composants
  - Un modèle par composants (*modèle local*)
    - bibliothèque de modèles réutilisables
  - *Composer* les modèles locaux en vue d'obtenir le modèle du système (*modèle global*)
    - besoin de définir des opérations de combinaisons des modèles
- Sur les automates, opérations de bases :
  - produit d'automates
  - composition synchrone

# Composition parallèle (synchrone) d'automates

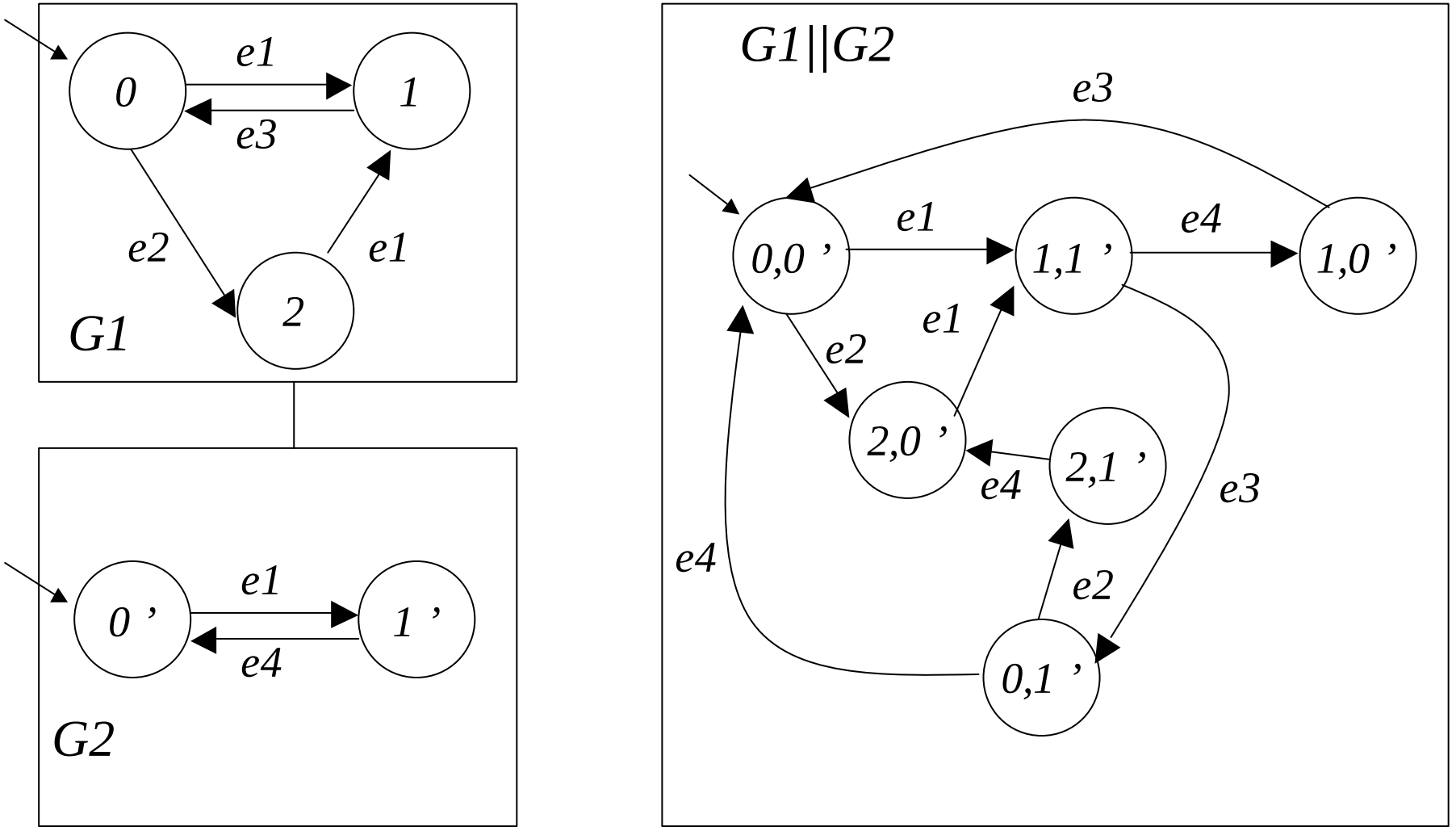
- $G=G1 \parallel G2$

- $G = Acc(X1 \times X2, E1 \cup E2, f, (x_{01}, x_{02}), X_{m1} \times X_{m2})$
  *avec*

  $f((x1,x2),e) = (f1(x1,e), f2(x2,e))$ si $e \in G1(x1) \cap G2(x2)$
  $f((x1,x2),e) = (f1(x1,e), x2)$ si $e \in G1(x1) \setminus E2$
  $f((x1,x2),e) = (x1, f2(x2,e))$ si $e \in G2(x2) \setminus E1$
  *indéfini sinon*

- Modèle de comportement associé à 2 composants: obtenu par composition synchrone sur les événements communs

# Exemple


0
1
e1
e3
e2
e1
2
G1
0 ’
1 ’
e1
e4
G2
G1||G2
e3
0,0 ’
e1
1,1 ’
e4
1,0 ’
e1
e2
2,0 ’
2,1 ’
e4
e3
e4
e2
0,1 ’

# Diagnostic à base de modèle de SED

- Étant donné un modèle,
- Étant donné des événements observables,
  - ✓ Flux d'observations arrivant en ligne:
    - ➢ Observations datées (réception) ou ordre partiel

Quels sont les événements non-observables qui « expliquent » (consistants) avec les observations ?

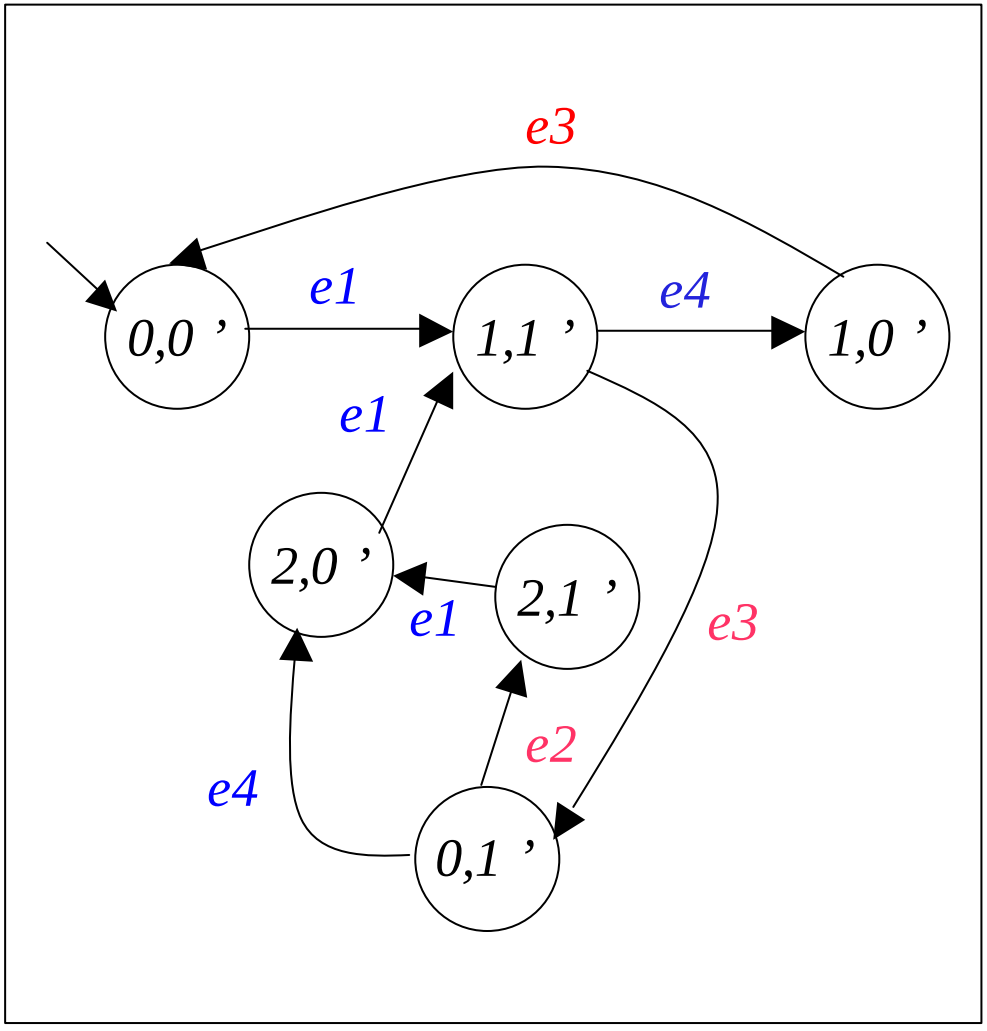
**Diagnostic** : Ensemble de trajectoires

- Événements de pannes (début / fin) + ordre partiel
- États possibles du système

Diagnostic $\Delta(O)$ = modèle global $||_{obs}$ O

synchronisation sur les événements observables obs

# *Exemple*




Événements de panne :

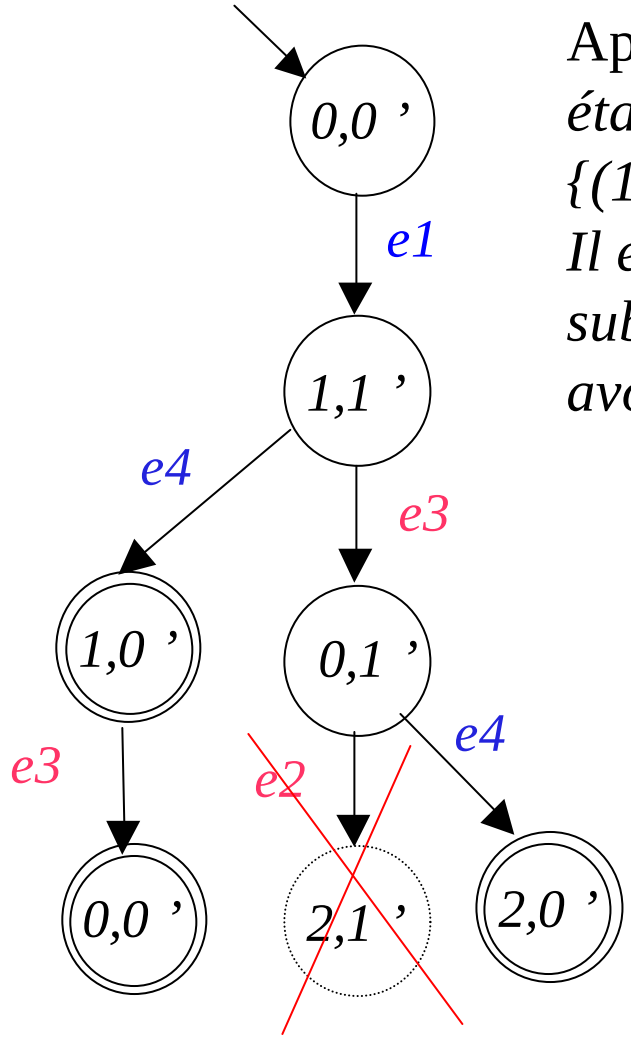
*e2, e3*

Événements observables :

*e1, e4*

Quel est le diagnostic si l'on sait que l'état initial du système est (0,0') et que l'on observe *e1* puis *e4* ?

# *Après l'observation de e1 puis de e4*

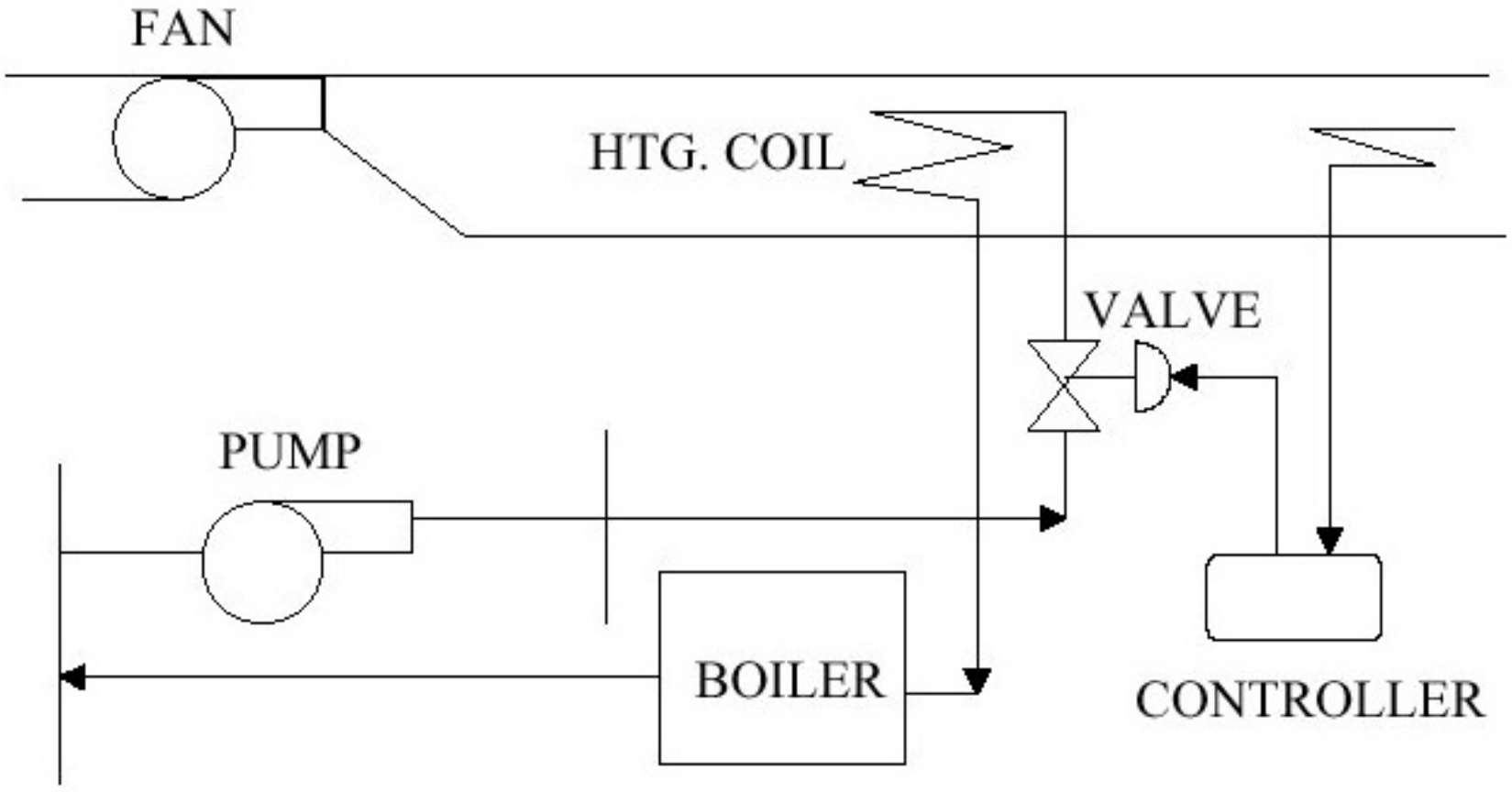



Après observation de *e1* et de *e4* :
*états possibles du système :*
*{(1,0'), (0,0'), (2,0')}*
*Il est possible que le système n'ait subi aucune panne. e2 n'a pas pu avoir lieu. e3 est possible.*

# Exemple : systeme d'air conditionné

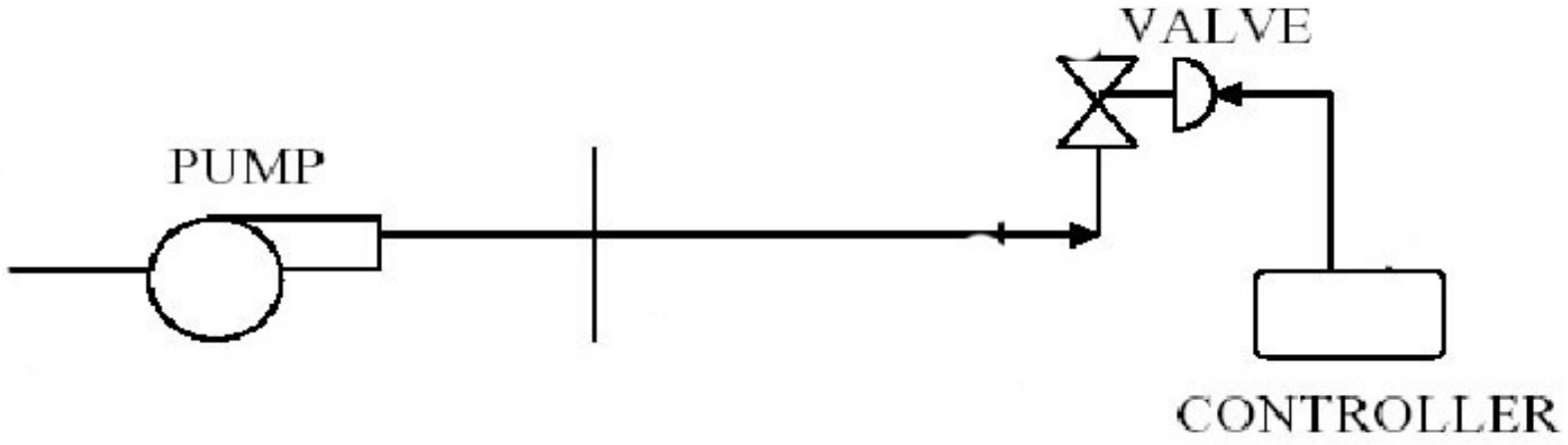

Stéphane Lafortune
*Dept of EECS, Detroit*
Meera Sampath
*Wilson Center for Research & Technology*

# HVAC system

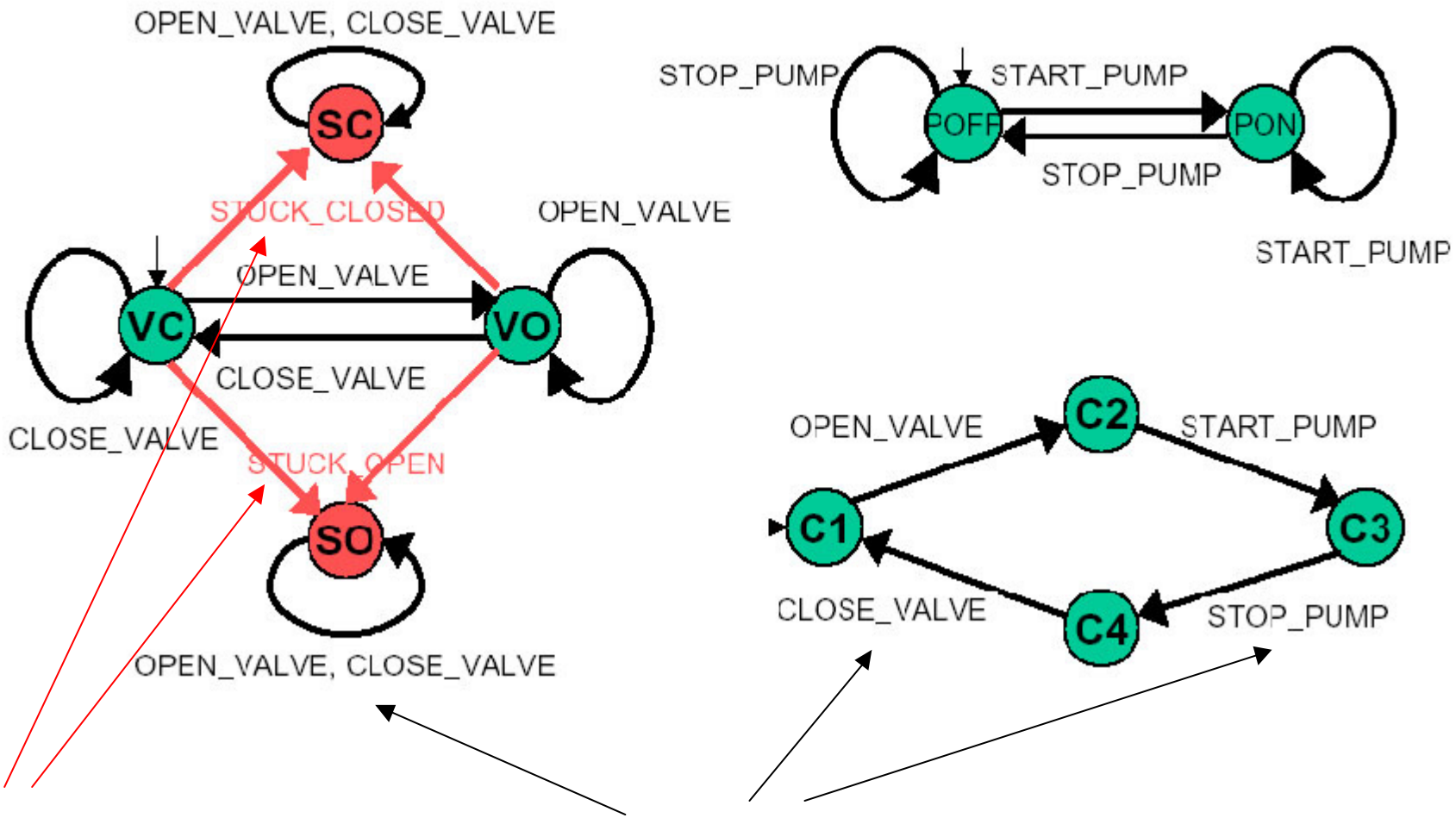


# *Une partie du système*

- **Capteur de pression**
  - **PP : « Pump Pressure »**
  - **NP : « No Pressure**
- **Capteur de flux d ’air**
  - **F : « Flow »**
  - **NF : « No Flow »**



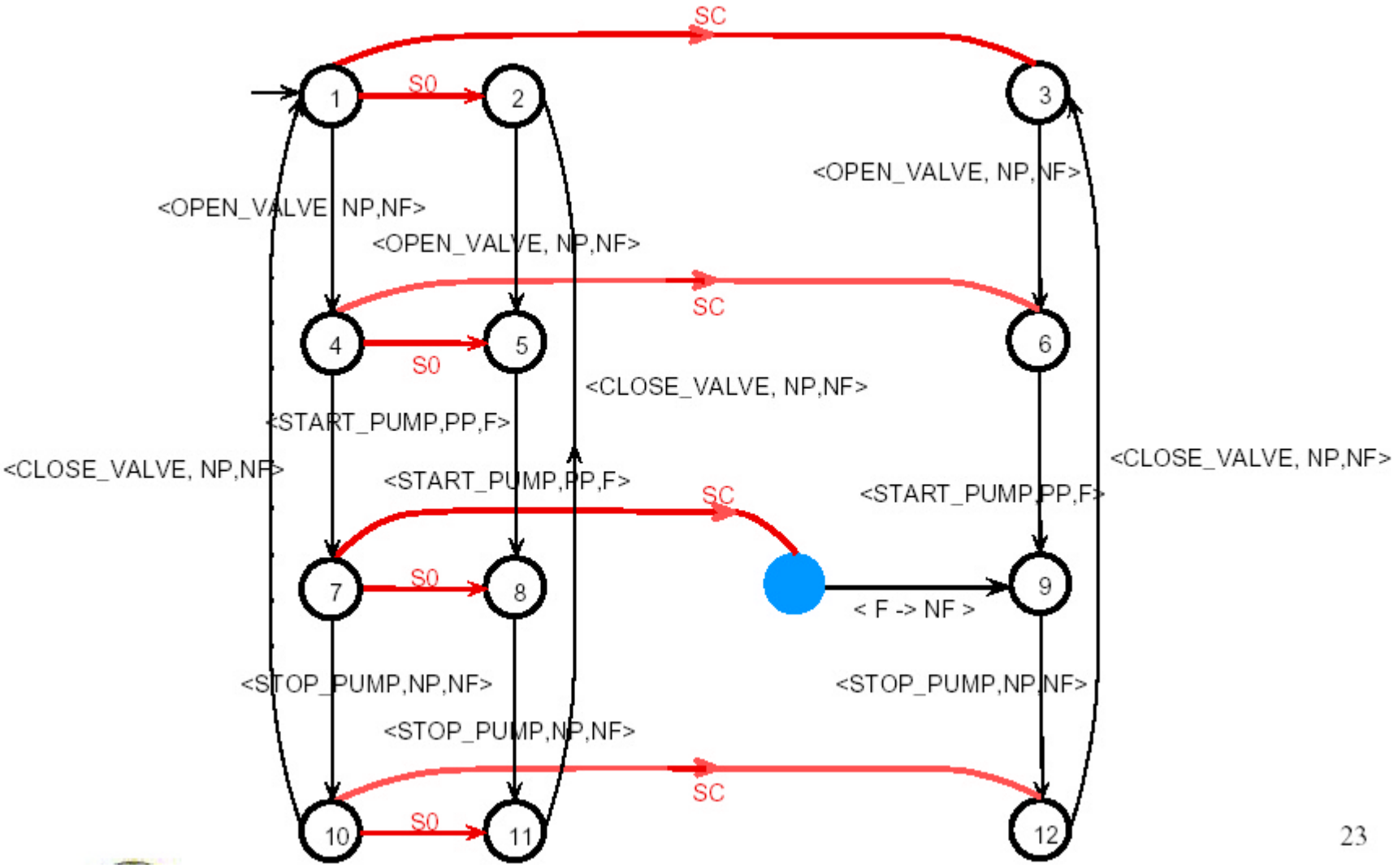
# Modèles de comportement des composants




pannes

actions de contrôle

# *Modèle global (après composition synchrone)*

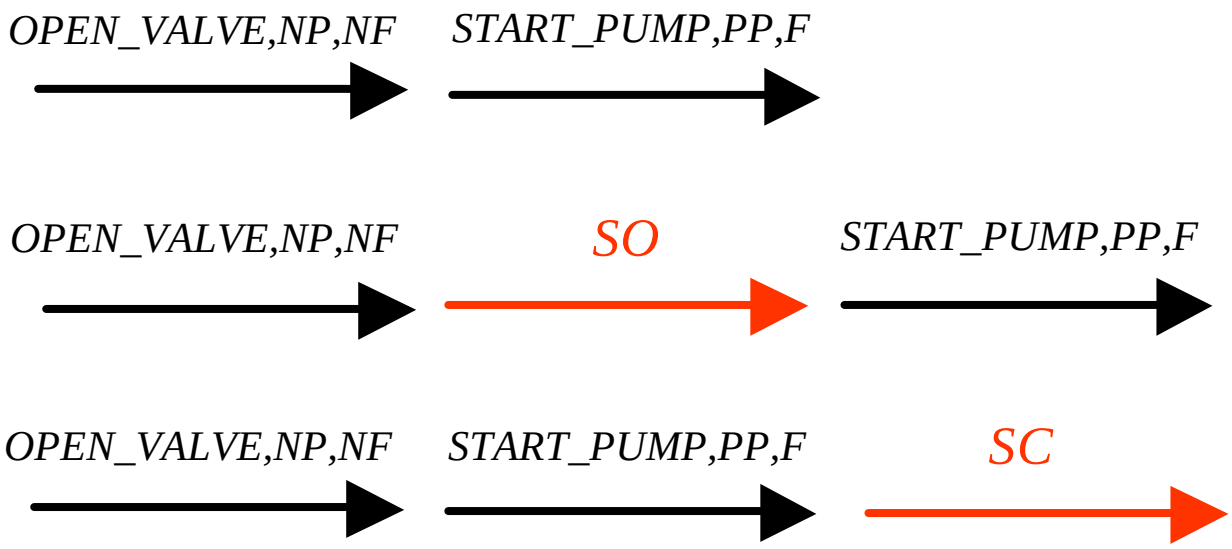


# *Résultat – voir automate*

Soit O une séquence d'événements observables :

*OPEN_VALVE,NP,NF*  *START_PUMP,PP,F*

$$D(O) = (1,4,7)(1,4,5,8)(1,4,7,9)(1,4,7,8)(1,2,5,8)$$



# Calcul du diagnostic : différentes *approches possibles*

- Approche simulation [Baroni et al.]
  - « tirer » les transitions en fonction des observations
    - complexe, méthode hors-ligne

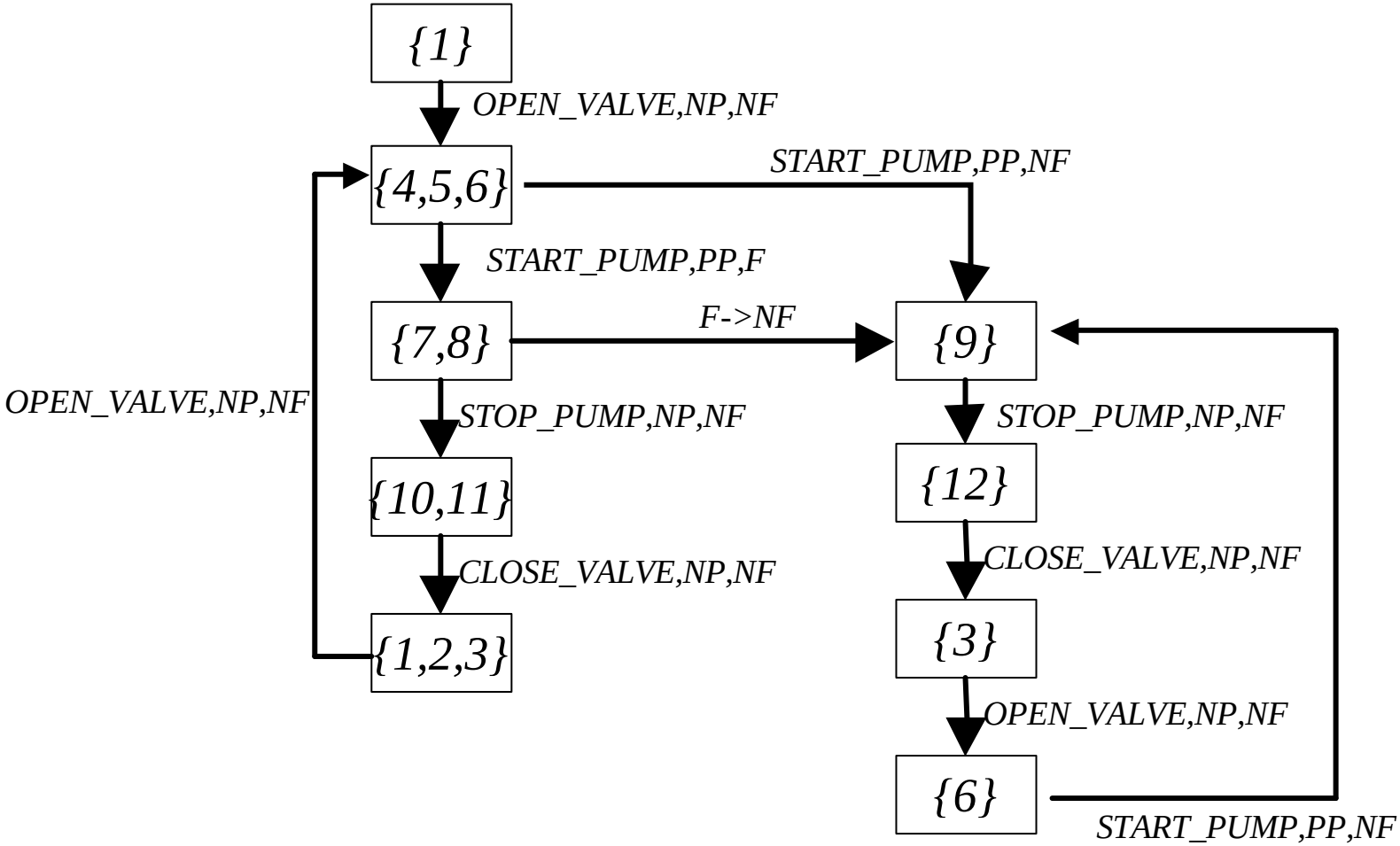
- Approche diagnostiqueur [Sampath et al.]
  - Compilation de l'information de diagnostic
    - plus efficace en-ligne

# *Automate observateur / diagnostiqueur*

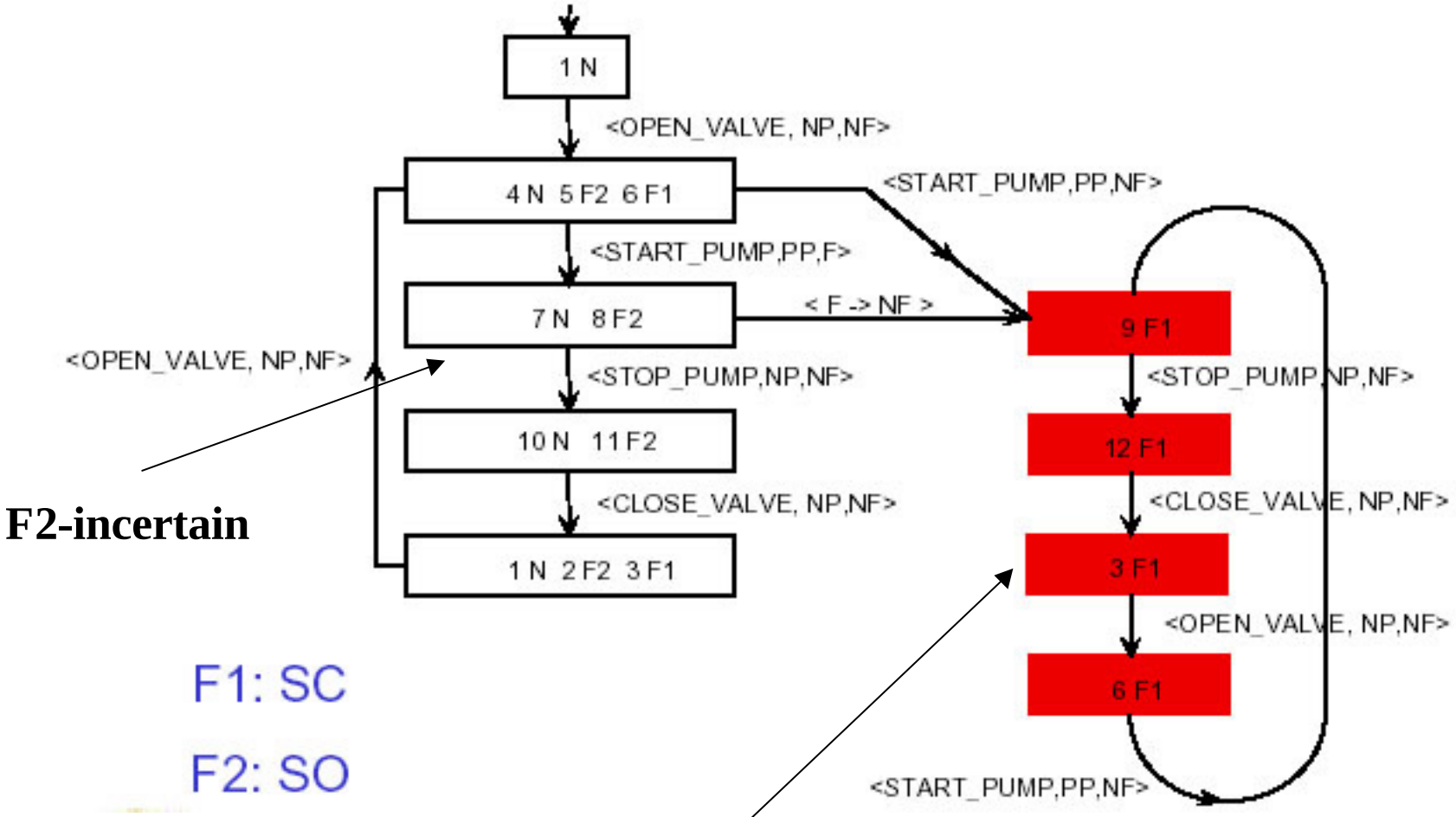
- Automate observateur : ens. des *comportements observables* du système
  - Construction par ε-réduction de l'automate du modèle global (avec ε : événement non-observable)
  - information de diagnostic : uniquement des états du système, pas d'information sur les événements de pannes

- **Diagnostiqueur** [Sampath et al. 95] : observateur **renseigné**
  - Etats du diagnostiqueur :
    - Candidats au diagnostic sous forme de couples (état,étiquette)
    - Étiquette : liste de pannes sur le chemin menant dans cet état
  - Etat qualifié de :
    - Normal : tous les candidats sont étiquetés par N
    - Fi-certain : tous les candidats ont Fi dans l'étiquette
    - Fi-incertaine : sinon

# *Observateur de HVAC*



# *Diagnostiqueur du HVAC*



# Di

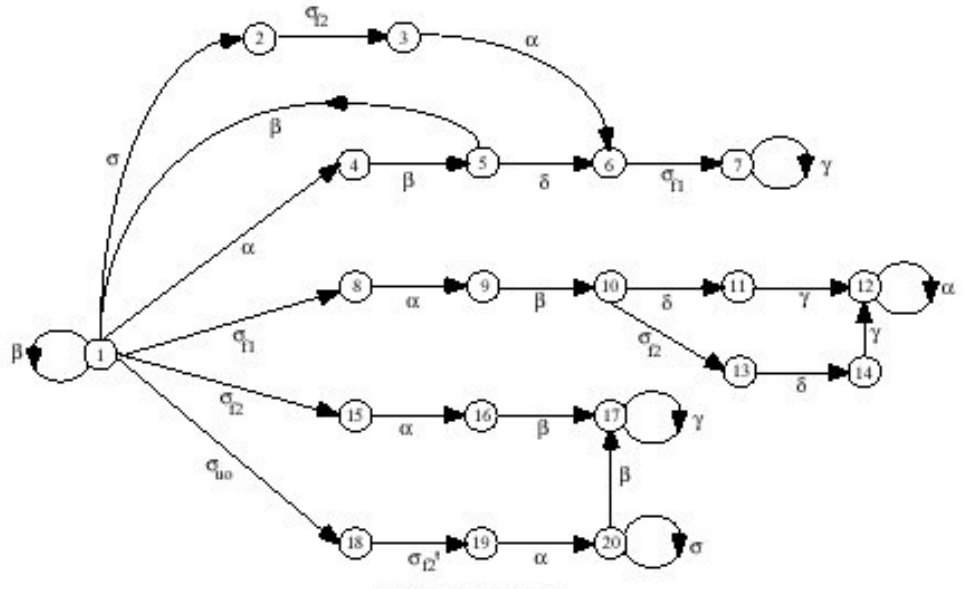



**Figure 3.5:** Example illustrating construction of the diagnoser $G_d$

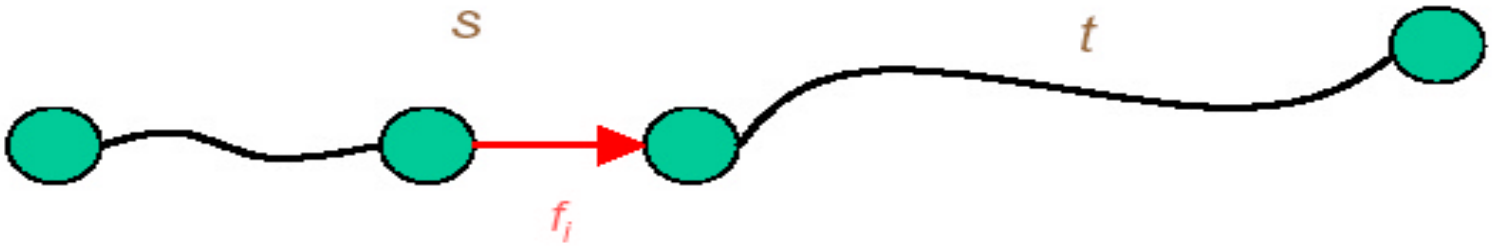
# *Diagnosticabilité de SED*

# *Introduction informelle*

- Partition $\Pi f$ des événements de panne :
  - Chaque événement de panne appartient à une *classe de panne (type)*
- Séquence O d'observations

- Un SED est *diagnosticable* s'il est possible de détecter toute *occurrence d'un type de panne* de $\Pi f$ en un *temps fini*, en s'appuyant *uniquement* sur les observations O

# Caractéristique d'un SED diagnosticable




- Trajectoire *s* terminée par un evt de panne *fi*
- Trajectoire *t* continuation de *s* suffisamment longue
- Chaque trajectoire « ressemblant » à *s.t* doit contenir une occurrence de panne du même type que *fi*
  - ressemblance : comportement observable

# *Définition formelle*

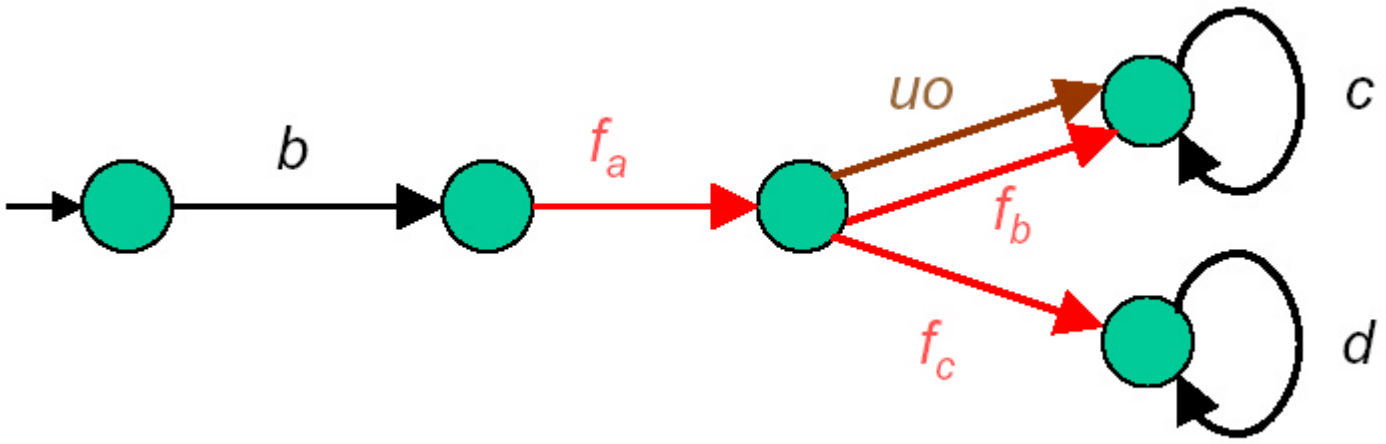
- Un langage L « prefix-closed » et vivant est diagnosticable relativement à une projection *P* et une partition Π*f* sur les evts *Ef* si :

$$( \forall\, i \in \Pi_f ) \; ( \exists\, n_i \in N ) \quad ( \forall\, s \in \Psi(E_{fi}) )$$
$$( \forall\, t \in L/s ) \qquad [\, \| t \| \geq n_i \Rightarrow \; D \,]$$

avec D : *condition de diagnosticabilité* $\omega \in P_L^{-1}[P(st)] \Rightarrow E_{fi} \in \omega$ .

- Projection *P* : enlève les événements non-observable d ’une trajectoire
- Projection inverse : $P_L^{-1}(y) = \{ s \in L : P(s) = y \}$
- Trajectoires se terminant par une panne de type *i :* $\Psi(E_{fi}) = \{ sa \in L : a \in E_{fi} \}$

# *Exemple*



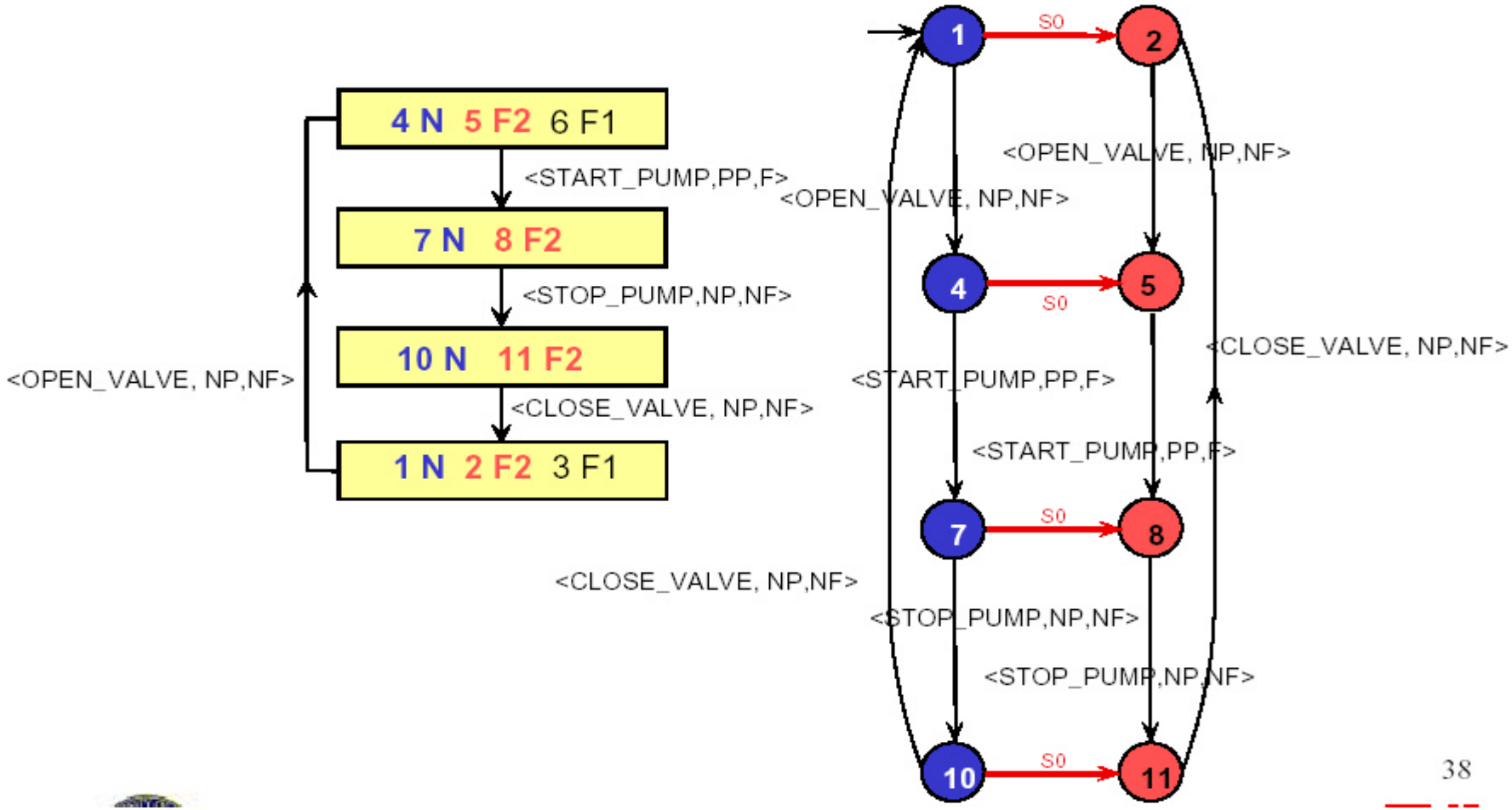

b,c,d : événements observables
uo : événement non observable
fa, fb, fc : événements de panne (non observables)

Si Πf = { {fa}, {fb}, {fc} } alors le système n'est pas diagnosticable
Si Πf = { {fa,fb}, {fc} } alors le système est diagnosticable

# *Diagnosticabilité et diagnostiqueur*

- Résultat formel :
  - un SED est diagnosticable si et seulement si son diagnostiqueur ne contient pas de *cycles indéterminés*
  - Cycle indéterminé :
    - cycle d'états Fi-incertains dans le diagnostiqueur
    - les états correspondants au cycle Fi-incertain forment aussi un cycle (observable) dans l'automate

# *Exemple*


4 N 5 F2 6 F1
<START_PUMP,PP,F>
7 N 8 F2
<STOP_PUMP,NP,NF>
10 N 11 F2
<CLOSE_VALVE, NP,NF>
1 N 2 F2 3 F1
<OPEN_VALVE, NP,NF>
1
S0
2
<OPEN_VALVE, NP,NF>
<OPEN_VALVE, NP,NF>
4
5
S0
<CLOSE_VALVE, NP,NF>
<START_PUMP,PP,F>
<START_PUMP,PP,F>
7
S0
8
<CLOSE_VALVE, NP,NF>
<STOP_PUMP,NP,NF>
<STOP_PUMP,NP,NF>
10
S0
11

# Cycle incertain mais pas indéterminé …

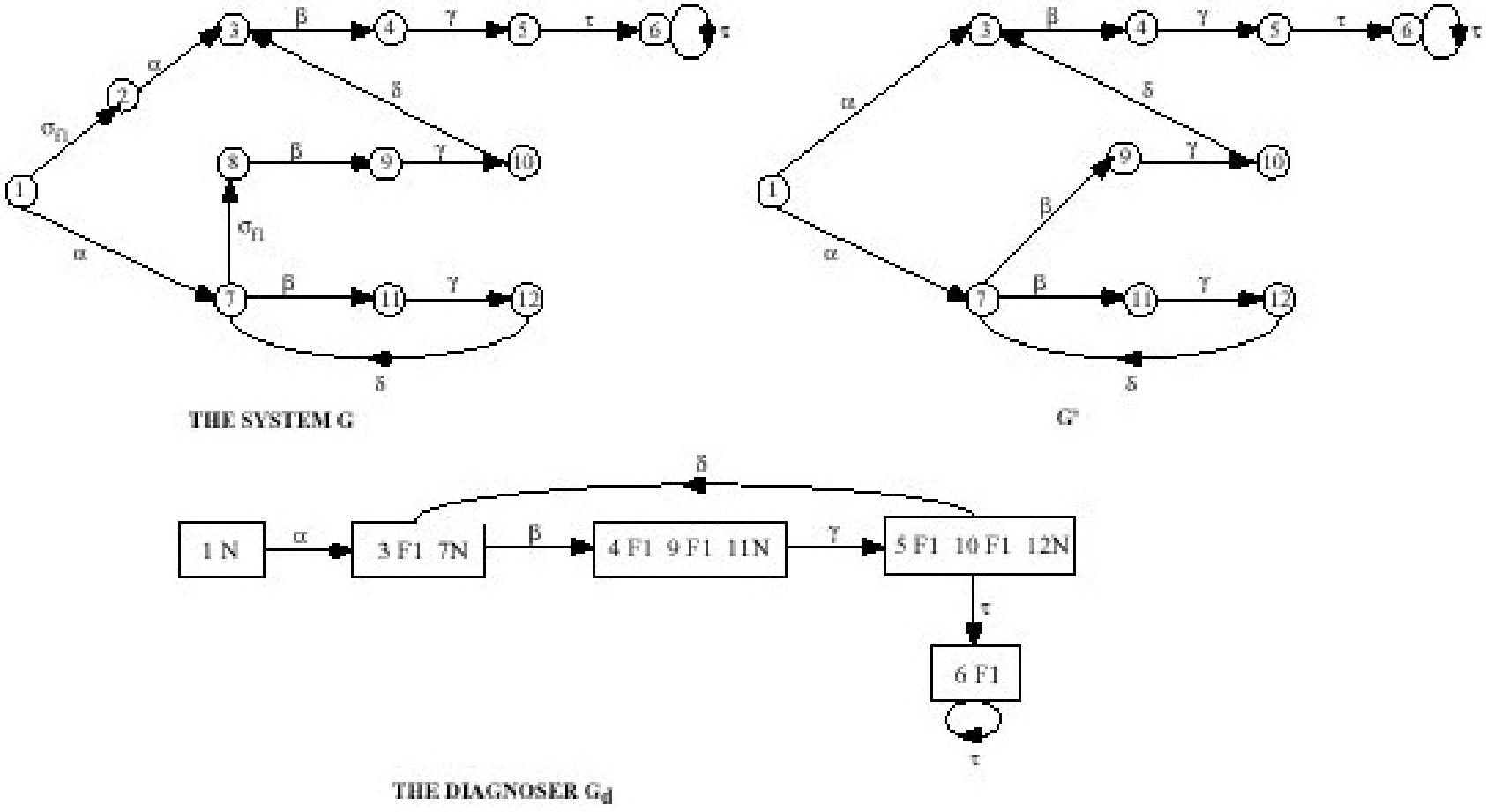



**Figure 3.7:** Example of a system with a cycle of $F_1$-uncertain states in its diagnoser $G_d$

# *Vérification de modèle et diagnostic*

# *Model-checking ?*

- Vérification automatique de systèmes complexes
  - langage de spécification de système
    - notion d'automates élémentaires
    - opération de composition
  - langage de spécification de propriété à vérifier : logique temporel
    - CTL, TCTL
  - algorithme efficace :
    - structures de données efficaces : BDDs
    - méthodes symboliques : ensemble de régions, graphe de simulation

Pourquoi ne pas utiliser les résultats du model-checking pour le calcul de diagnostic?

# *Application au diagnostic de SED*

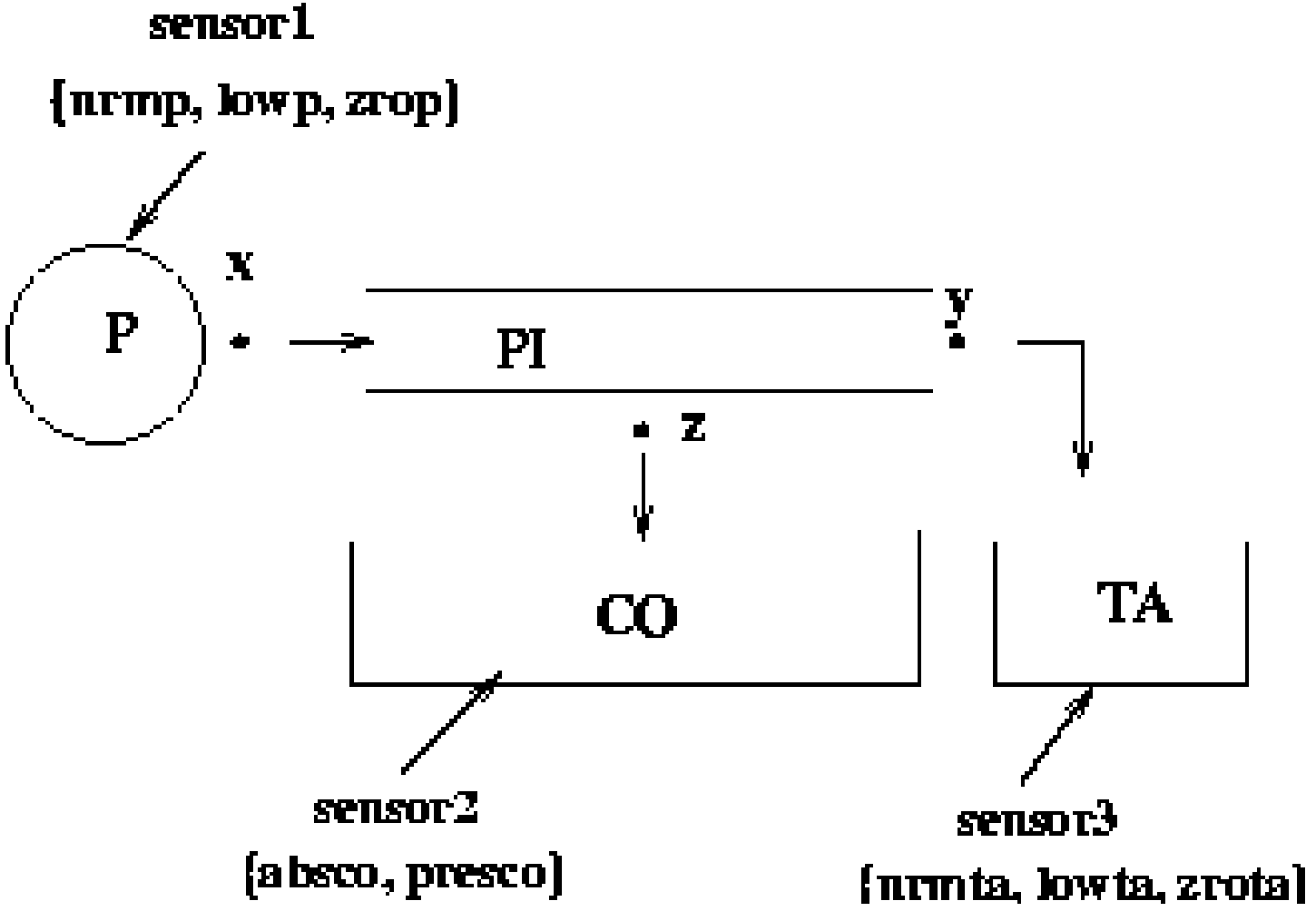
- Une propriété intéressante : l'atteignabilité
  - vérifier qu'une situation peut se produire, c'est rechercher des trajectoires dans le modèle qui vérifient cette situation
- Diagnostic : un problème d'atteignabilité
  - Soit OBS= [obs1, obs2] la séquence d'observations du système

$$KRONOS\ [A, Obs_1 \Rightarrow \exists\Diamond\ \ Obs_2]$$

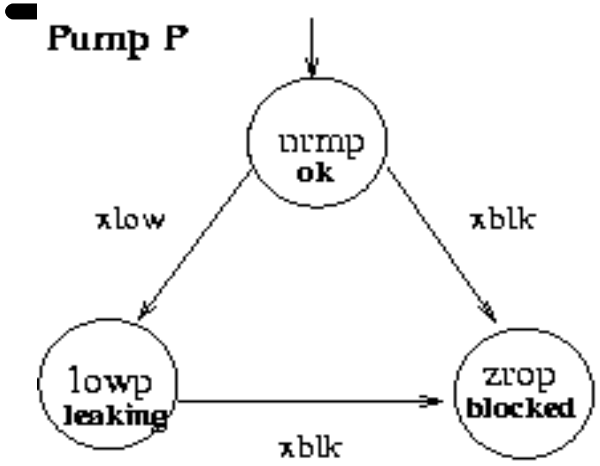
$\exists\Diamond\,\varphi$ : Il existe une trajectoire vers un état qui a la propriété $\varphi$

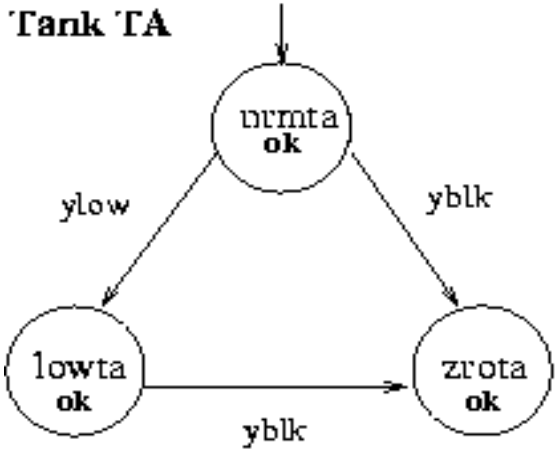
$\forall\Diamond\,\varphi$ : Pour toute trajectoire, il existe un état qui a la propriété $\varphi$

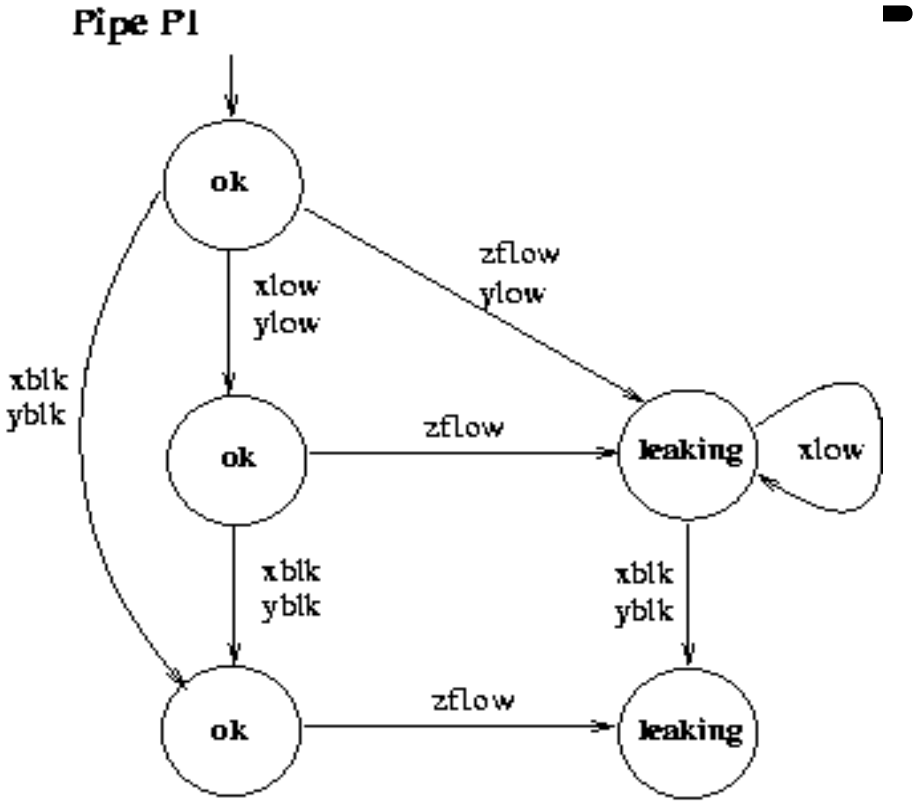
# *Un exemple*

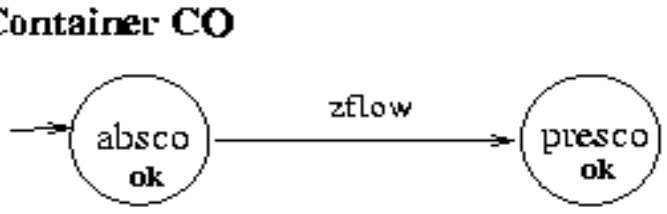

sensor1
{nrmp, lowp, zrop}
x
P
PI
y
z
CO
TA
sensor2
{absco, presco}
sensor3
{nrmta, lowta, zrota}

# *Modèle du système*



# *Formule de diagnostic*

Requête Kronos :

$$KRONOS\ \ [A, Obs_1 \Rightarrow \exists\Diamond \quad Obs_2]$$

Dans notre exemple :

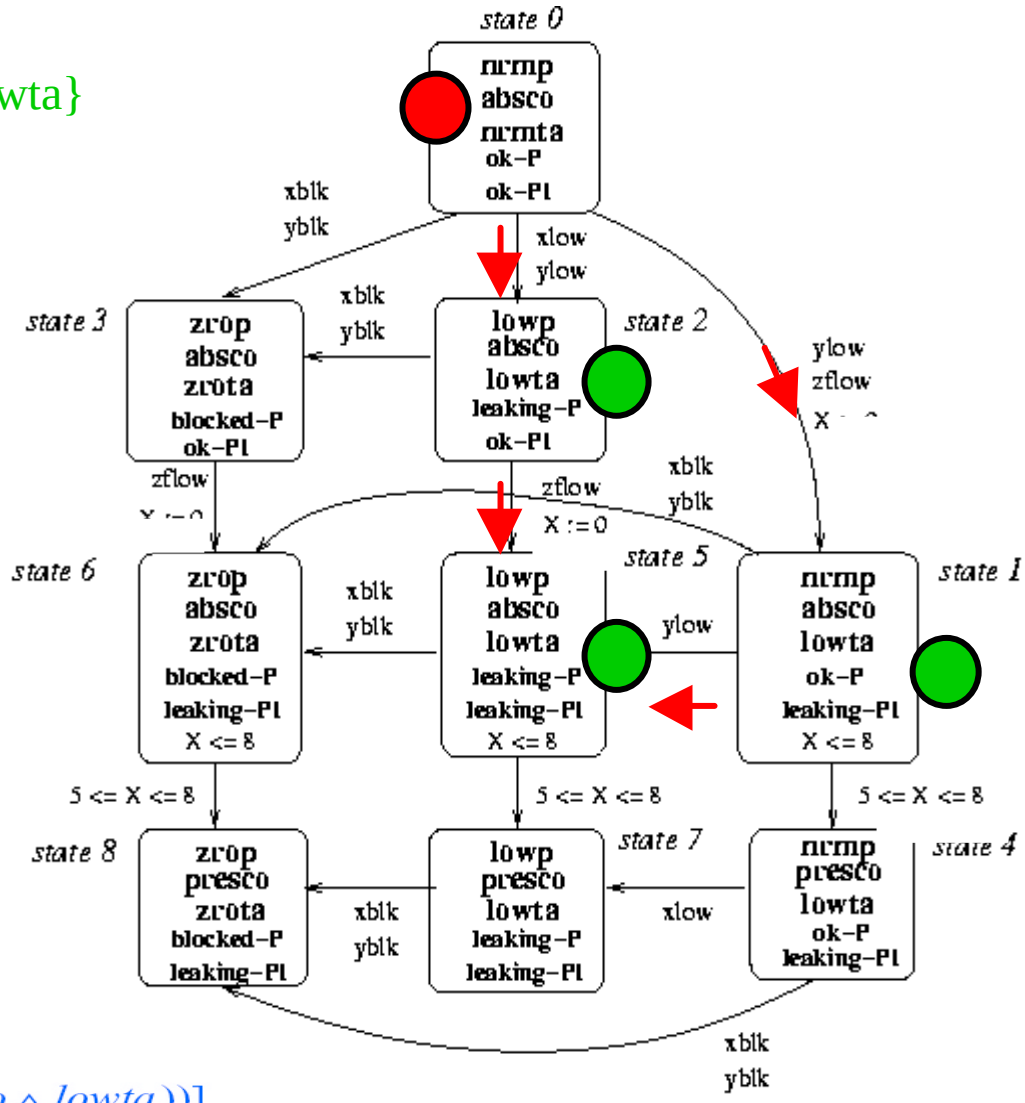
$$Kronos[A, nrmp \wedge absco \wedge nrmta \Rightarrow$$

$$\exists\Diamond(nrmp \wedge absco \wedge lowta) \vee (lowp \wedge absco \wedge lowta))]$$

# *Résultat du calcul d'atteignabilité*

Obs1= {nrmp,absco,nrmta}

Obs2={lowp,absco,lowta},{nrmp,absco,lowta}




$$Kronos[A, nrmp \wedge absco \wedge nrmta \Longrightarrow$$

$$\exists\Diamond(nrmp \wedge absco \wedge lowta) \vee (lowp \wedge absco \wedge lowta))]$$

# *Conclusion intermédiaire*

- Modélisation du comportement (automate)
  - Normal + pannes : détection + diagnostic
  - Contraintes temporelles : séquence – (+ contraintes quantitatives)
- Observations
  - Ensemble ordonné : séquence mais plutôt ordre partiel
- Diagnostics : trajectoires
  - Comportements compatibles avec les observations,
  - Définis par SD $\oplus$ OBS
- Algorithmes :
  - Analyse des transitions possibles (parcours de l'automate) et recherche des chemins compatibles
  - Automate réduit aux événements observables avec mémoire des événements de panne (par exemple diagnostiqueur)

# Conclusion intermédiaire : DES: Problèmes et approches proposées

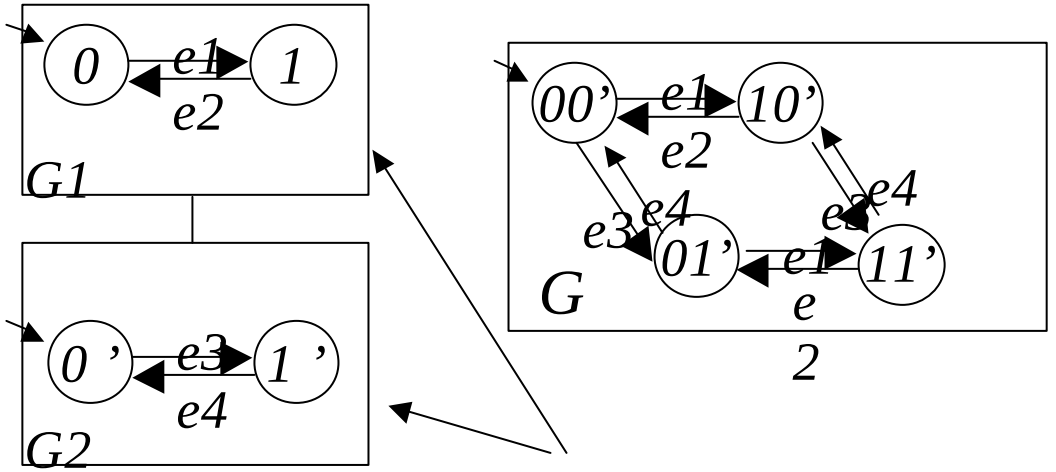
- **Modèles** :
  - Taille du modèle : approches « décentralisées »
  - Acquisition du modèle
    - Récupération des spécifications (UML) ?
    - Langage à base de règles / automate
  - Changement de la topologie au cours du diagnostic : « reconfiguration »
- **Observations** :
  - incertaines, incomplètes
  - Diagnosabilité et choix d'implantation des capteurs
- **Efficacité des algorithmes** :
  - Exécuter certaines tâches en parallèle de manière répartie
  - Stratégie s'appuyant sur un critère de sélection : rech. heuristique
  - Représentation plus compacte des trajectoires
    - Réduction d'ordre partiel, BDD
  - Outils de type model-checking

# Conclusion intermédiaire : DES: Problèmes et approches proposées

- Modèles :
  - **Taille du modèle : approches « décentralisées »**
  - Acquisition du modèle
    - Récupération des spécifications (UML) ?
    - Langage à base de règles / automate
  - Changement de la topologie au cours du diagnostic : reconfiguration
- Observations
  - incertaines, incomplètes
  - Diagnosabilité et choix des meilleurs capteurs
- Efficacité des algorithmes:
  - Exécuter certaines tâches en parallèle (de manière répartie)
  - Stratégie s'appuyant sur un critère de sélection (rech. heuristique)
  - Représentation plus compacte des trajectoires
    - Réduction d'ordre partiel, BDD
  - Outils de type model-checking

# *Approches « décentralisées »*

- Modèle global du système =
  - Composition des modèles de ses composants SD = ⊗SDi et donc explosion du nombre d'états et du nombre de trajectoires, en particulier pour les composants « concurrents »



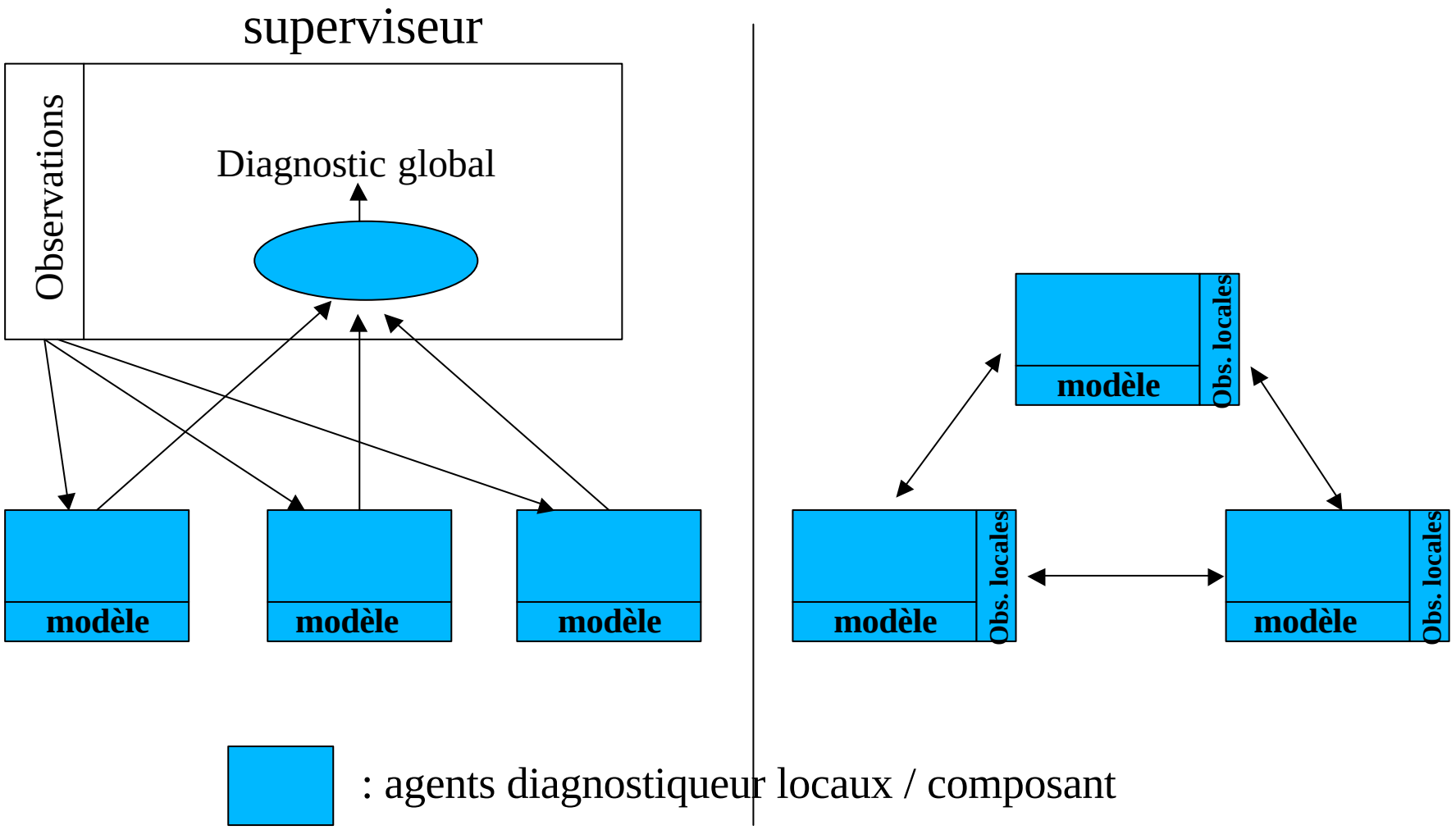

modèles locaux

- Idée :
  - Observations locales
  - Diagnostics locaux
  - Modèles locaux (composants)

# Approches « décentralisées »
## 1 – superviseur ou non ?

- Deux architectures :
  - Superviseur en charge du diagnostic global
    - Diagnostic décentralisé avec décision centralisée
      - Capteur global (capteurs locaux + canaux de communication vers le superviseur)
      - Synchronisation, calcul, mémorisation des diagnostics par le superviseur
    - Approche multi-agents avec tableau noir
  - Pas de superviseur mais échanges entre composants
    - Diagnostic (pas de diag. global) et décision « distribués »
      - Pas de capteur global mais des capteurs locaux
      - Synchronisation, calcul, mémorisation des diagnostics en local
      - Arrêt (+ moments de synchronisation)
    - Approche multi-agents communicants

# *Deux architectures*




: agents diagnostiqueur locaux / composant

Quel modèle pour les agents diagnostiqueurs?

# « décentralisées »
# 2 – modèle global ou local ?

- Les agents partagent le modèle global
  - Agent : modèle global / observations « locales »
    - diagnostic global calculé de manière répartie

- Les agents ne disposent que d'un modèle local
  - Agent : modèle local / observations « locales » :
    - diagnostic local
  - Synchronisation des diagnostics locaux :
    - Par le superviseur qui synchronise les diagnostics locaux et calcule et mémorise le diagnostic global.
    - Sans superviseur :
      - Les agents synchronisent leurs diagnostics en communicant leurs trajectoires aux autres agents
      - Chaque agent mémorise les diagnostics locaux qui le concernent
      - Détection de la fin ?

# « décentralisées »
# 3 – Quand fait-on la synchronisation ?

- Synchronisation : Le diagnostic local est-il compatible avec les observations non visibles par l'agent ? Satisfait-il les contraintes de synchronisation?
  - Élimination des diagnostics locaux non satisfaisants
  - Construction des diagnostics + en + globaux si possible (si superviseur)

- Quand synchroniser ?
  - À chaque observation
  - Sur des fenêtres temporelles regroupant un ensemble d'observations

- Un point dur :
  - Quand peut-on être sûr qu'un diagnostic local ne pourra pas être synchronisé (ordre partiel entre les observations locales) ?
  - Utilisation d'informations sur les délais max; notions de fenêtres sûres (les canaux de communication sont vides); diagnostic avec vision à + x observ.

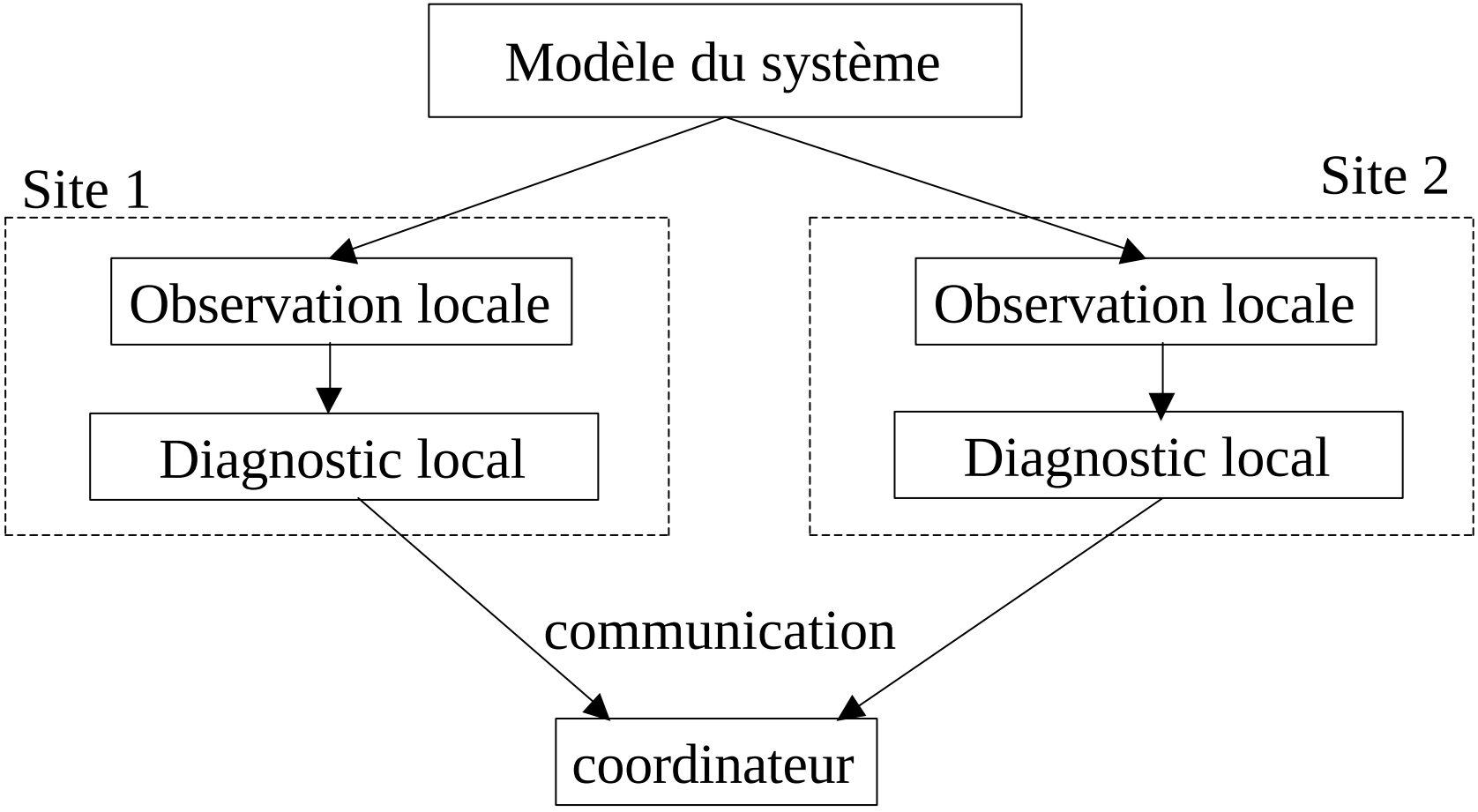
# *Trois approches*

- Approche avec modèle global + superviseur : Lafortune, Debouk, Sengupta et al.
  - Trois protocoles proposés (protocole 1)
- Approche sans superviseur + modèle local : Fabre et al.
  - Modèle : réseaux de Petri ou règles (pièces) + probabilités
  - Algorithme de calcul de type Viterbi
  - Structure de donnée partagée : « hook »
- Approche avec superviseur + modèle local : Pencolé et al.
  - Modèle : automates communicants
  - Algorithme avec diagnostiqueur + réduction d'ordre partiel
  - Fenêtre d'observation et stratégie de synchronisation (les plus interagissants d'abord)
- Voir approche avec superviseur + modèle local (hors-ligne) : Zanella et al.

# Approche Lafortune et al.

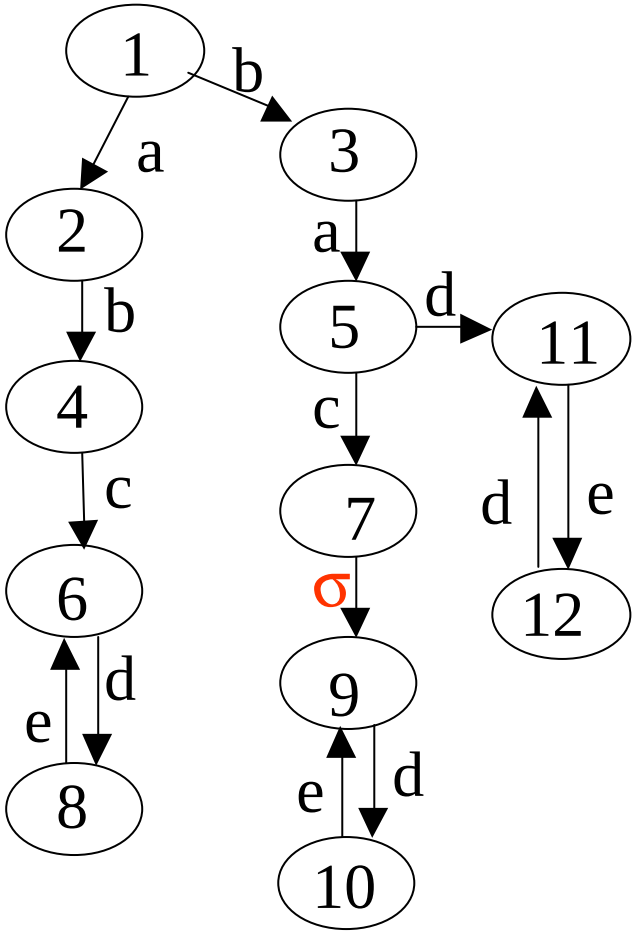
« A coordinated decentralized protocol for failure diagnosis of discrete event systems » R. Debouk, S. Lafortune and D.Teneketzis, The University of Michigan, Ann Arbor, MI 48109-2122

Proposition d'un protocole permettant de calculer les mêmes diagnostics qu'en version centralisée : protocole 1

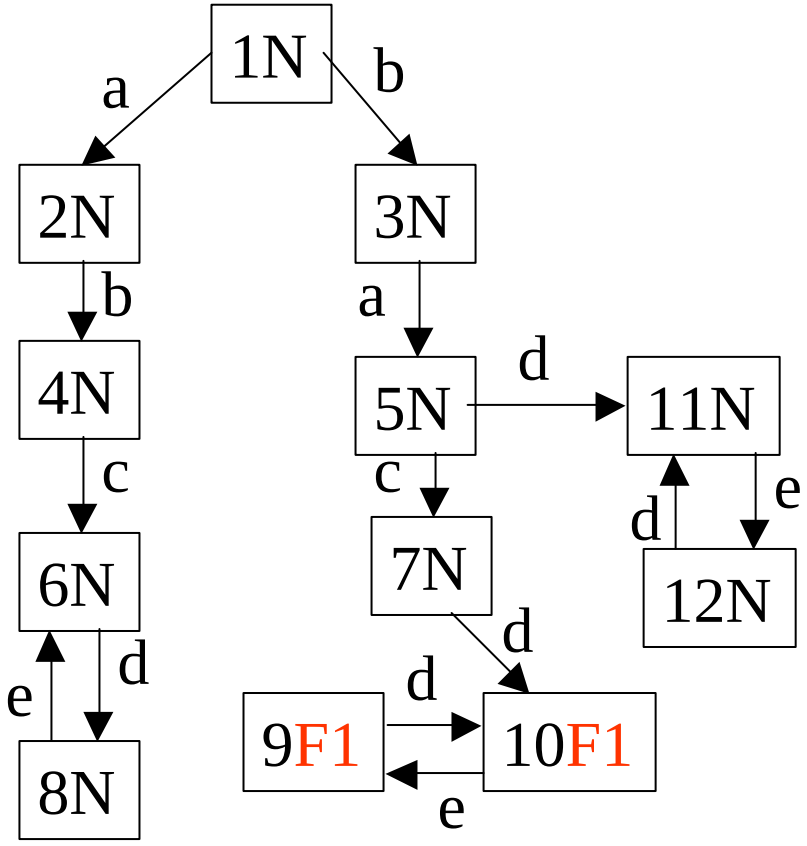


# Exemple

$\Sigma_f = \Sigma_{uo} = \{\sigma\}$




$G_d$ : diagnostiqueur



# Diagnostiqueur étendu + « unobservable reach »

**Diagnostiqueur étendu :**

$Ge_d = (Qe_d, \Sigma_o, \delta e_d, qe_0)$

q ? $Qe_d$ est de la forme $\{((x_1, k_1),(y_1, l_1)) , ... ,((x_n, k_n),(y_n, l_n))\}$

$qe_0 = \{((x_0, N),(x_0, N))\}$ où $y_1, ..., y_n$ sont les états du modèle global dans lesquels peut se trouver le système, $l_1, ..., l_n$ les étiquettes correspondantes, et $x_i$ correspond au prédécesseur direct de $y_i$ dans G' et $k_i$ son étiquette associée.

**« Unobservable reach » :** (i et j représentent les 2 sites avec i,j ? {1, 2} et $i \neq j$, q est un état du diagnostiqueur étendu pour le site i)

Idée : le site i, qui est dans l'état q, prévoit ce que le site j peut observer grâce aux évènements observables par j seul.

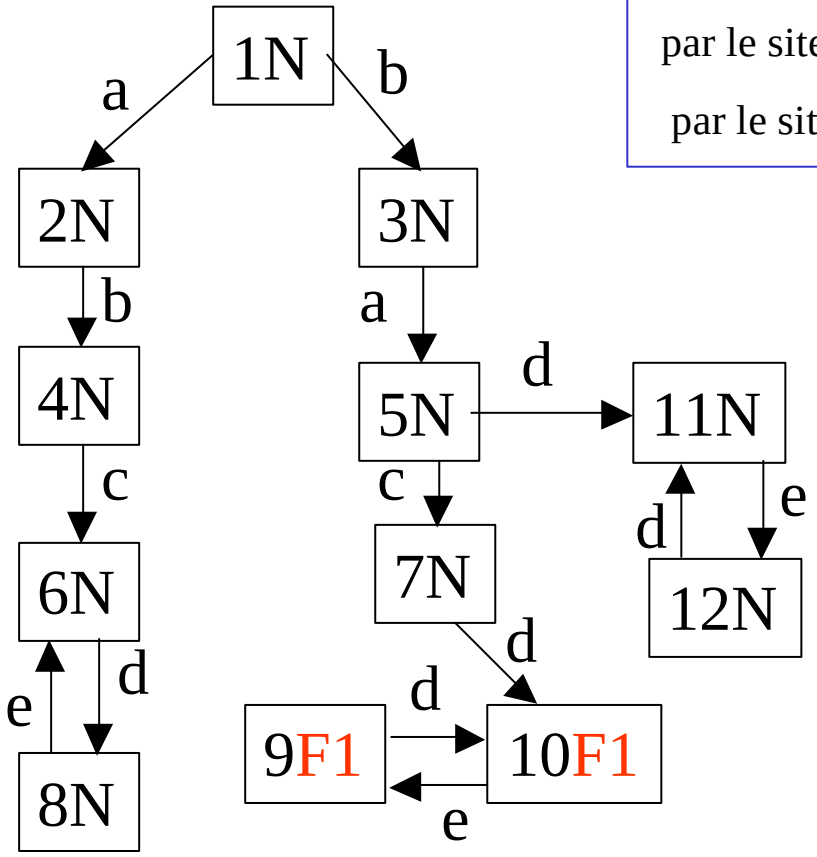
– ajouter à l'état q les états atteignables à partir de q, par une séquence d'événements observables uniquement par le site j.

# Diagnostiqueur étendu

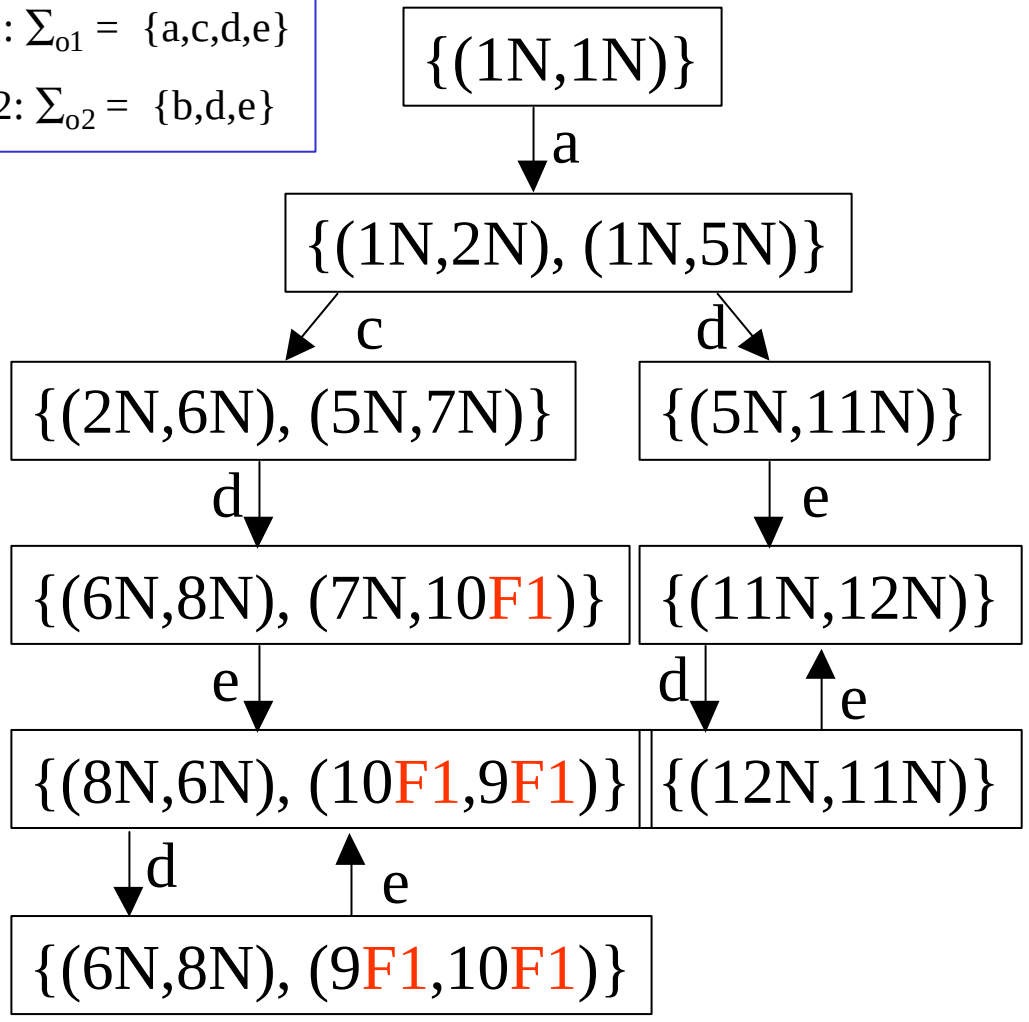
évènements observables :

par le site 1: $\Sigma_{o1}$ = {a,c,d,e}

par le site 2: $\Sigma_{o2}$ = {b,d,e}




*Diagnostiqueur global*



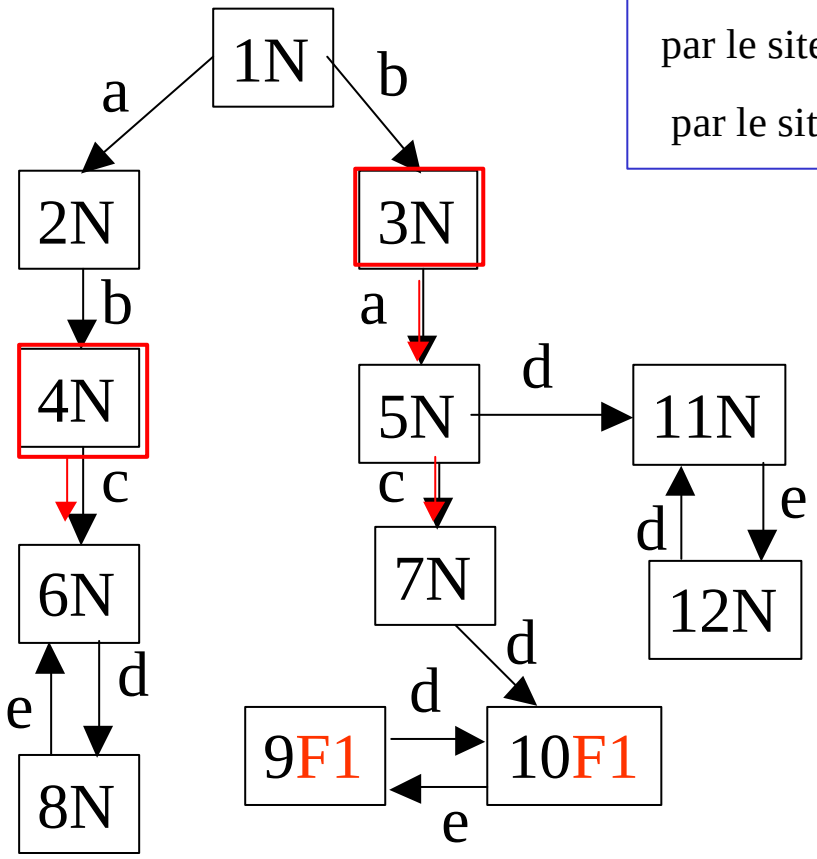

*Diagnostiqueur étendu pour le site 1 ($Ge_{d1}$)*

# « unobservable reach »

évènements observables :

par le site 1: $\Sigma_{o1}$ = {a,c,d,e}

par le site 2: $\Sigma_{o2}$ = {b,d,e}




Pour q = {(1N,3N),(1N,4N)}

$UR_2(q)$ = {(1N,3N),(1N,4N),(3N,5N),(5N,7N),(4N,6N)}
par {},{},{a},{a,c},{c}

***Diagnostiqueur global***

# La communication entre les sites et le coordinateur

- passage d'information du site i vers le coordinateur quand un événement x est observé par le site i :

  - L'état courant q du diagnostiqueur étendu pour le site i (après changement d'état du à l'observation de x).
  - $UR_i(q)$
  - $SB_i$ : un bit égal à 1 si x ? $\sum_{oj}$ et à 0 sinon. ($j \neq i$)

# Le coordinateur

9 registres :

- C (resp. $C_{old}$) est l'« information » courante (resp. précédente) du coordinateur .
- SB est un bit qui est mis à 1 lorsque que le dernier événement observé par un site est aussi observable par l'autre.
- R1 (resp. R2) est le dernier état reçu du site 1 (resp. 2).
- R3 (resp. R4) est le dernier « unobservable reach » reçu du site 1 (resp. 2).
- $SB_{1old}$ (resp. $SB_{2old}$) est la sauvegarde du dernier bit $SB_1$ (resp. $SB_2$) reçu du site 1 (resp. site 2).

Deux opérations d'intersection pour les ensembles d'états du diagnostiqueur étendu : $q_1 \cap_{ei} q_2$ où i ? {L,R} et $q_1 \cap_c q_2$

6 règles pour faire évoluer le diagnostic :

lorsque un événement est observé par le site 1 :

| SB | $SB_1$ | C | New SB |
|---|---|---|---|
| 0 | 0 | $(R1 \cap_{ei} R4) \cap_c C_{old}$ | 0 |
| 0 | 1 | attendre | 1 |
| 1 | 1 | $(R1 \cap_{ei} R2) \cap_c C_{old}$ | 0 |

# Conclusion sur l'approche ~~Debouk et al.~~

- approche décentralisée avec superviseur :
  - qui gère une certaine synchronisation
- les deux sites doivent connaître :
  - le modèle global
  - les observables de l'autre site

# Approche Fabre et al.

*Fabre-Benveniste-Jard-Haar / Projet Magda / 2000*

- Pas de superviseur / Modèle local + capteurs locaux
- Etant donné une observation et un ensemble de trajectoires, chaque agent peut utiliser le modèle local pour étendre les trajectoires compatibles avec ses observations et éliminer les autres (Ext et Red)
  - Algorithme de type Viterbi (modèle probabiliste)
- Structure de données : « hook » pour décrire une trajectoire (une transition + pointeur vers la précédente)
  - ***H*** : (index, état final, coût, dernière transition, pointeur vers le précédent) avec index : nombre d'observables expliqués

# Approche Fabre et al. Approche modèle global sans superviseur

**Les agents ont chacun accès au modèle global et savent qui doit traiter les observations**

**Algorithm 3:** two players (player $i$ described)

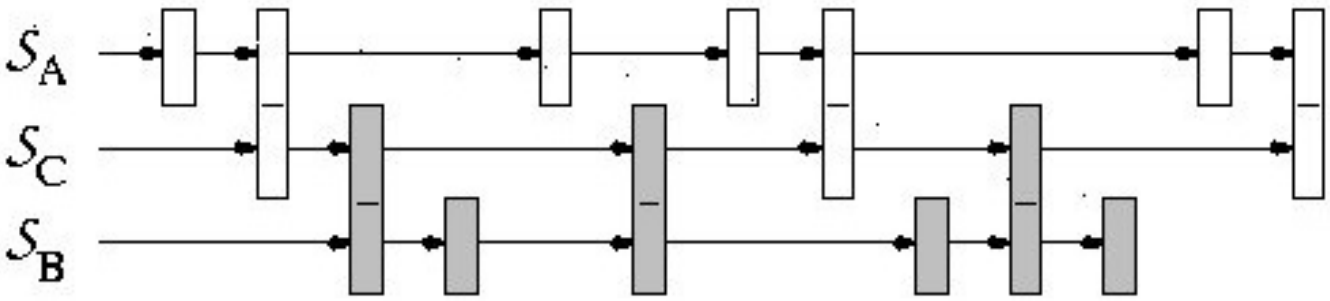
1. initialization
   a. $\mathcal{A} := \{(0, s_0, 0, \emptyset, \emptyset)\}$
   b. $\mathcal{A}_i := \mathsf{Type}_i(\mathcal{A})$
2. forward sweep : until `global-end` is detected
   a. on decision to process local hooks
      - $\mathcal{A}'_i := \mathcal{A}_i$ (memory)
      - $\mathcal{A}_i := \mathsf{Vit}_i^{k_i}(\mathcal{A}_i)$ for some integer $k_i$
      - send $\mathsf{Type}_j(\mathcal{A}_i \setminus \mathcal{A}'_i)$ to player $j$ (if non empty)
   b. on reception of a message $\mathcal{M}$ from player $j$
      - $\mathcal{A}_i := \mathcal{A}_i \oplus \mathcal{M}$
   c. `local-end` := all hooks in $\mathcal{A}_i$ are finished
3. backtrack

$\mathsf{Type}_i(A)$ : sélectionne les trajectoires relevant de l'agent i

**U ÅV = Red(U È V)**

# Approche modèle local sans superviseur

- Synchronisation par partage de ressources :
  - Modèles locaux SA et SB + modèle SC ne traitant que des événements partagés entre A et B
    - (hA, hB, hC) : « hooks » propres à A et B + hooks partagés
    - L'agent A (resp.B) utilise hA (resp.hB) et hC



# Approche modèle local sans superviseur

- Calcul des trajectoires par A :
  - L'agent A transforme (hA,hC) en (hA',hC) si il utilise un événement non partagé ou en (hA',hC') si il utilise un événement partagé
- Communication :
  - L'agent A envoie à B les « hooks » de hC qu'il a transformés en hC'(hC -> hC') dans $C_A$ (étendus par A et en attente de B) et ceux qui ont été traités dans $F_A$
  - Mise à jour des trajectoires tenant compte des événements synchronisés
- Mémorisation des trajectoires locales : $W_A \cup A$
  - $A_A$ : trajectoires à étendre par A
  - $W_A$ déjà traités par A et donc enlevés de $A_A$ mais en attente de B. Si traités par B, elles sortent de $W_A$, soit pour $A_A$, si étendues par B, soit définitivement sinon

**Algorithm 5** two players, distributed knowledge (player $A$ described)

1. initialization

   a. $h_A^0 := (0, a_0, 0, \emptyset, \emptyset)$
   b. $h_C^0 := (c_0, \emptyset, \emptyset)$
   c. $\mathcal{A}_A := \{(h_A^0, h_C^0)\}$
   d. $\mathcal{W}_A := \emptyset$

2. forward sweep : until **global-end** is detected

   a. on decision of processing hooks
      - select $\mathcal{A}'_A \subset \mathcal{A}_A$
      - $(\mathcal{N}_A, \mathcal{C}_A) := \mathsf{Ext}_A(\mathcal{A}'_A)$
      - $\mathcal{N}_A := \mathsf{Red}_A(\mathcal{N}_A)$
      - $\mathcal{F}_A := \mathsf{Fin}(\mathcal{A}'_A, \mathcal{N}_A, \mathcal{A}_A)$
      - $\mathcal{A}_A := (\mathcal{A}_A \setminus \mathcal{A}'_A) \oplus \mathcal{N}_A$
      - $\mathcal{W}_A := \mathcal{W}_A \oplus \mathcal{A}'_A$
      - send $(\mathcal{C}_A, \mathcal{F}_A)$ to player $B$

   b. on reception of message $(\mathcal{C}_B, \mathcal{F}_B)$ from player $B$
      - $\mathcal{A}_A := \mathcal{A}_A \oplus \mathsf{Update}(\mathcal{A}_A, \mathcal{C}_B) \oplus \mathsf{Update}(\mathcal{W}_A, \mathcal{C}_B)$
      - remove from $\mathcal{W}_A$ all pairs $(h_A, h_C)$ such that $h_C \in \mathcal{F}_B$

   c. **local-end** = all hooks in $\mathcal{A}_A$ are finished

3. backtrack



**U ÅV = Red(U È V)**

# *Approche Fabre et al.*

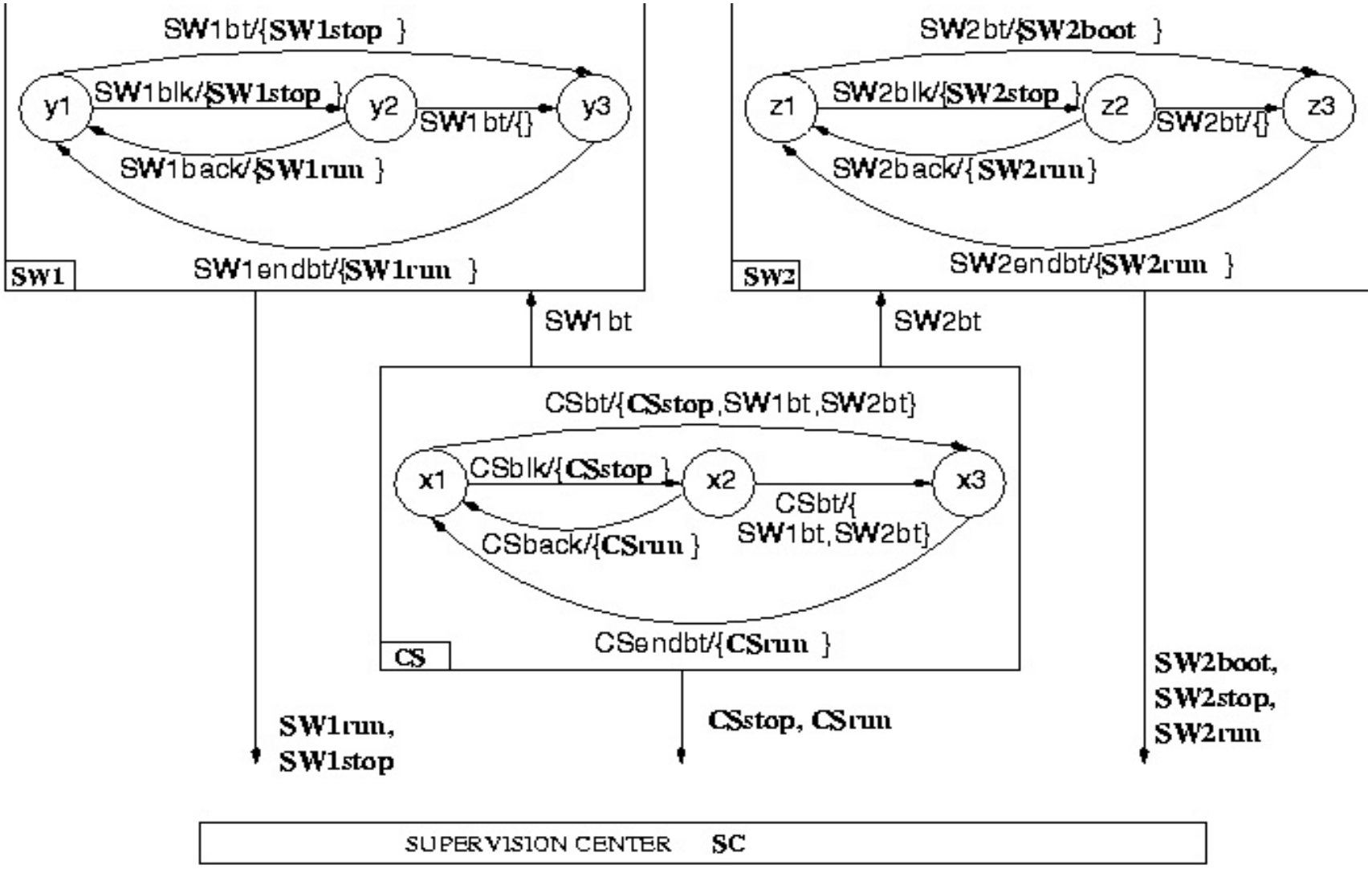
- Expérimenté dans le cadre de Magda
- Conditions sur la structure du système : arbre
- Extension à N joueurs ?

# *Approche Pencolé et al.*

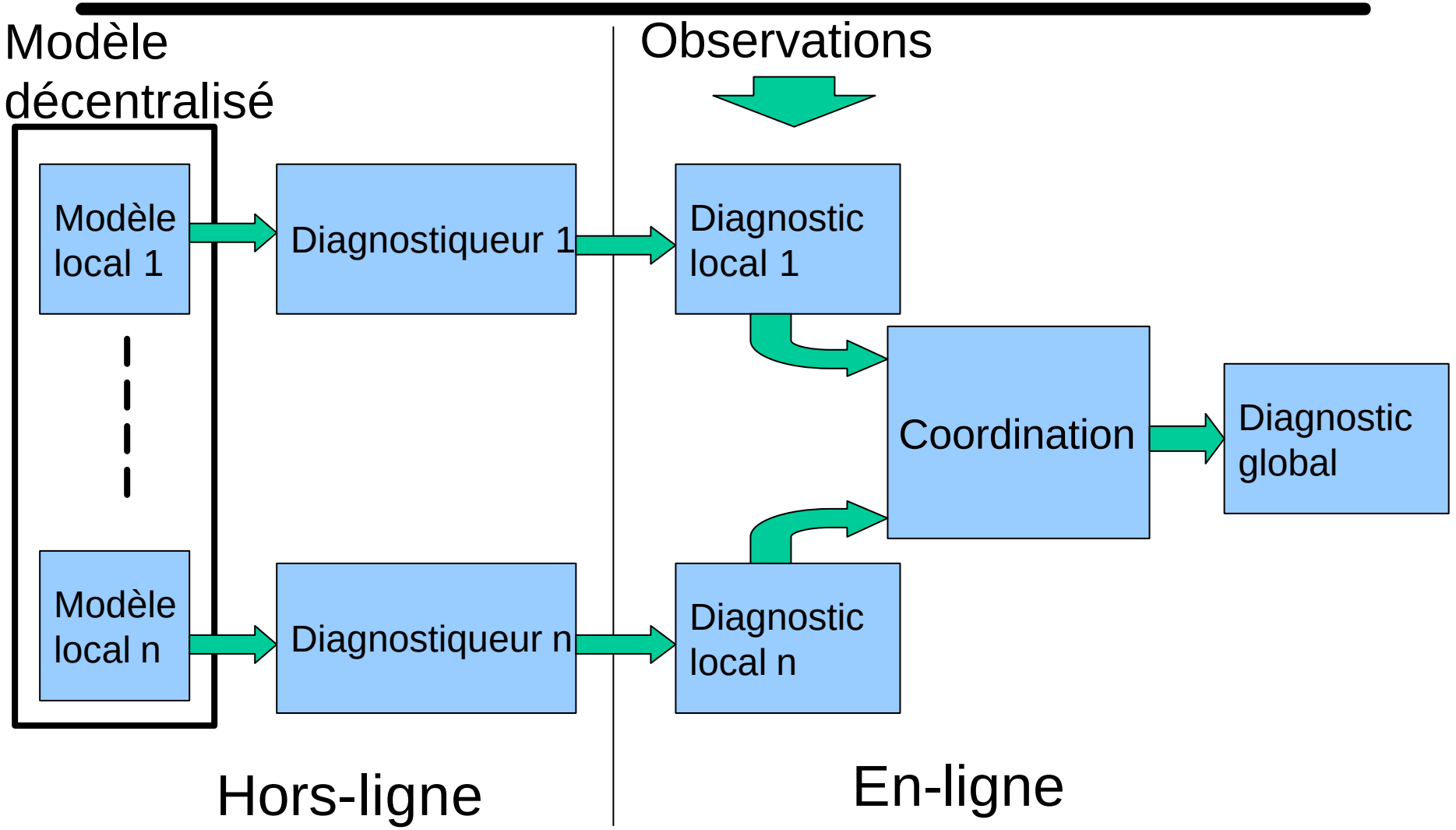
- Modèle local (automates communicants) + superviseur
- Canaux de communications : files
  - Ordre sur les observations reçues => ordre sur les observations émises par chaque composant
- Diagnostics locaux
  - Sous forme d'automates communicants

    (+ réduction d'ordre partiel)
  - Calculés par diagnostiqueur
- Stratégie de synchronisation
  - À la fin d'une fenêtre d'observations (sûre)
  - En vue de construire un diagnostic global (+ élimination)
  - Basée sur les interactions entre diagnostics locaux
    - Pas de synchronisation (évite des produits cartésiens inutiles) si pas d'interactions
    - Ordre des synchronisations favorisant les plus interagissants

# *Exemple: un réseau de télécommunications*

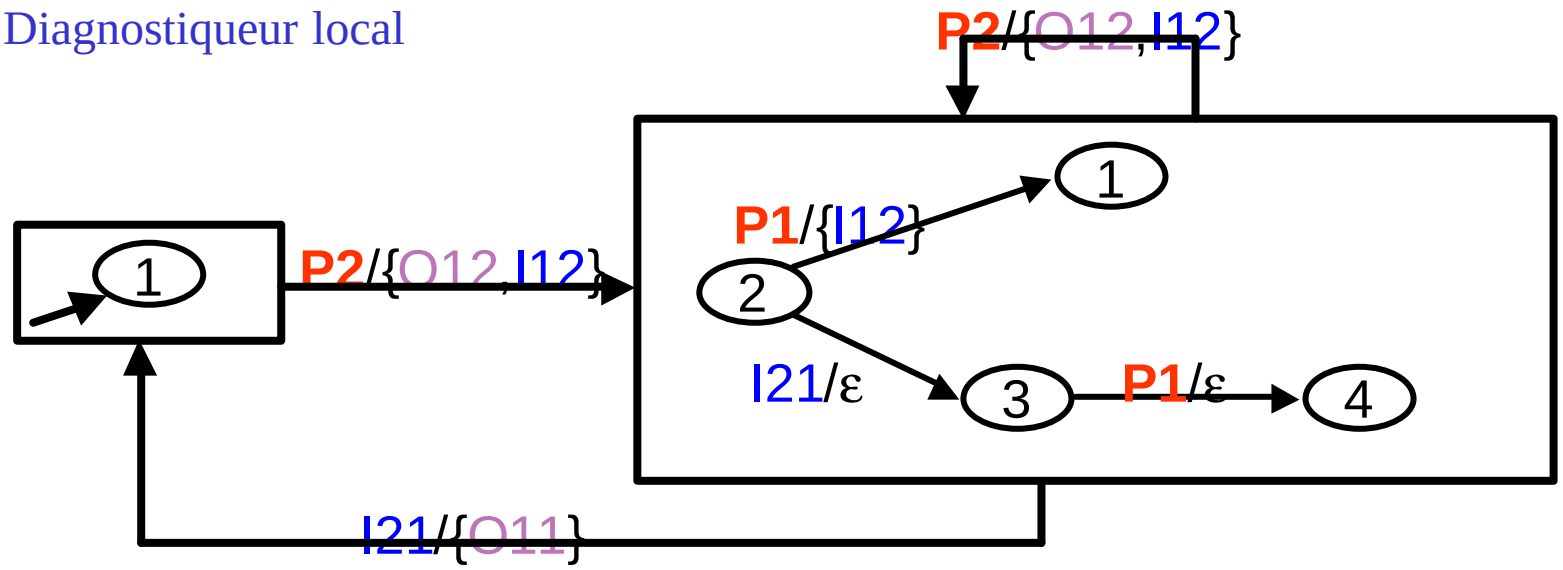


# *Une approche décentralisée*

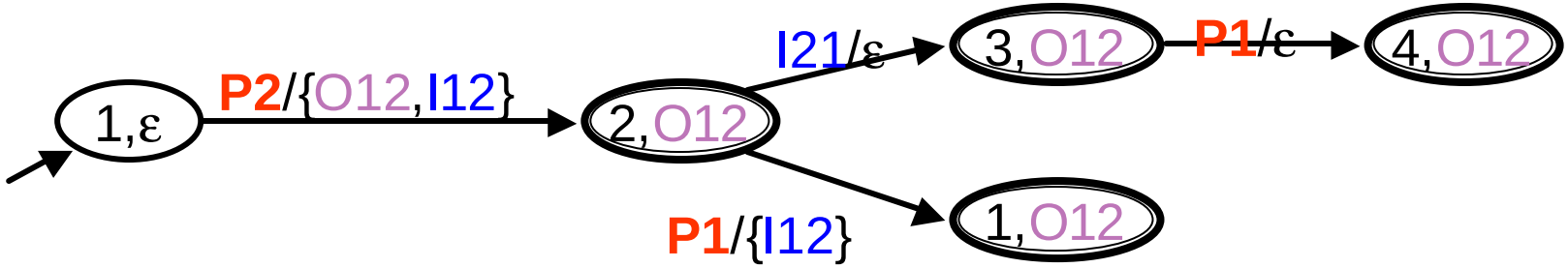


# *Diagnostiqueur et diagnostic local : exemple*

Diagnostiqueur local




Diagnostic local pour l'observation o12 : dépliage - I21 et I12 sont des messages entre le composant 1 et le composant 2 (à synchroniser)



# *Des diagnostics locaux au diagnostic global*

- Fusion des diagnostics locaux
  - élimination des événements internes
  - vérification des contraintes (synchronisation entre l'émission et la réception d'un événement interne)

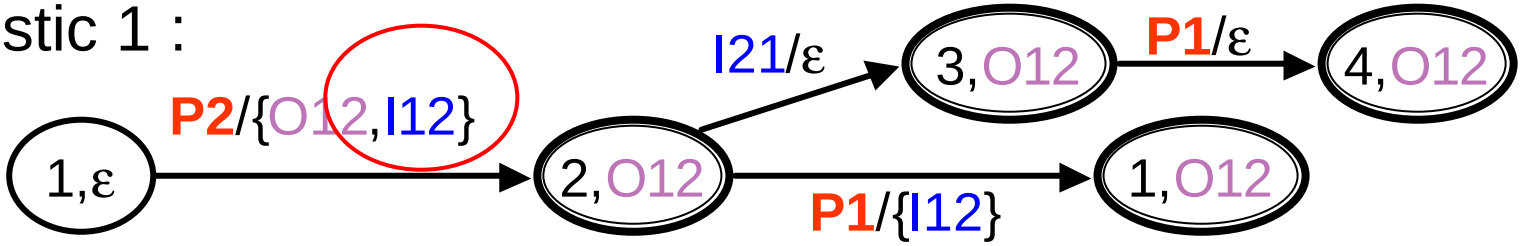
- Fusion : composition d automates (noté $\bigcirc$)

$$\Delta((x_1,\ldots,x_n),\{s_1,\ldots,s_n\}) = \bigcirc_{i=1}^{n} \Delta_i(x_i, s_i)$$

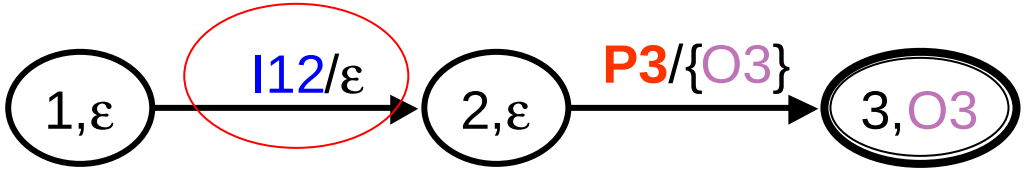
**états initiaux** observations

# *Fusion des diagnostics locaux*

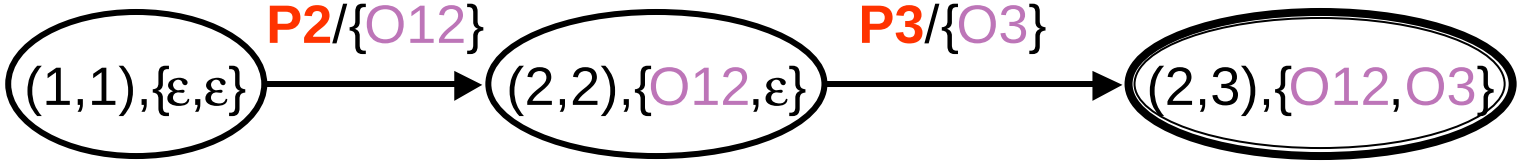
Diagnostic 1 :




Diagnostic 2 :

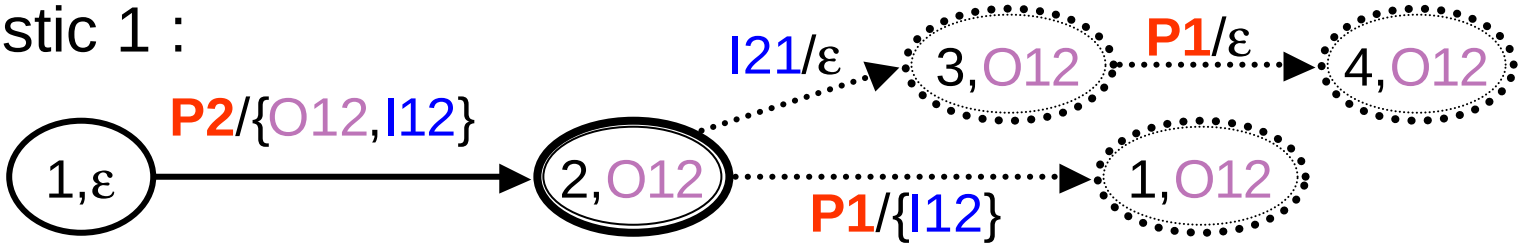



Fusion des 2 diagnostics :

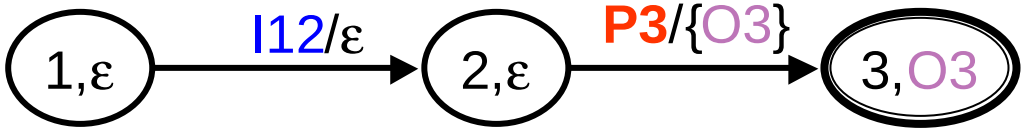


# *Fusion des diagnostics locaux (2)*

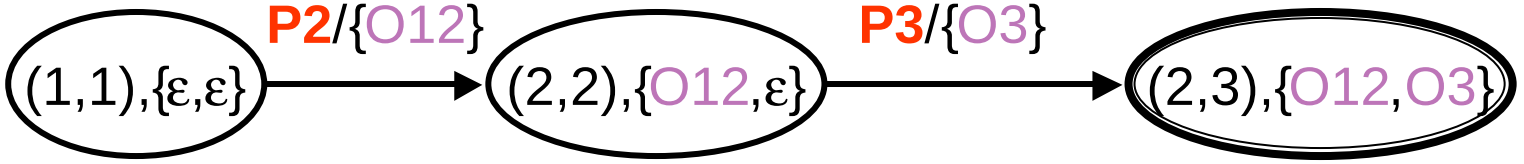
Diagnostic 1 :




Diagnostic 2 :




Fusion des 2 diagnostics :



# *Stratégie de coordination*

- Composition
  - Opération commutative et associative
  - Certains ordres d'opérations sont plus efficaces

- Stratégie
  - Favoriser le parallélisme dans la coordination
  - Composer en premier les diagnostics qui interagissent le plus

# *Exemple de fusion*

Quatre diagnostics locaux :

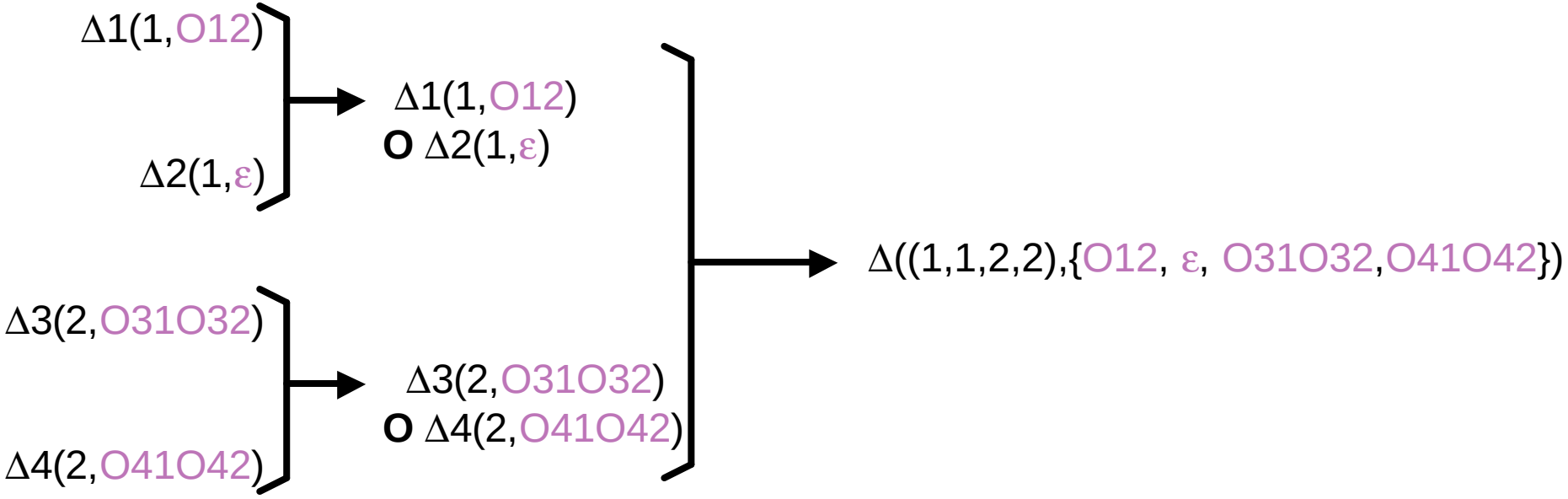
$\Delta_1(1,O12)$, $\Delta_2(1,\varepsilon)$, $\Delta_3(2,O31O32)$, $\Delta_4(2,O41O42)$

Diagnostics interagissants :

$\Delta_1(1,O12)$ et $\Delta_2(1,\varepsilon)$

$\Delta_3(2,O31O32)$ et $\Delta_4(2,O41O42)$

Fusion :



# *Fenêtre sûre*

- Fenêtre sûre :
  - Ensemble d'observations telles que toute observation émise par un composant lors de la réaction a été reçue
    - Canaux de communication vides
  - Permet d'éliminer les diagnostics locaux qui ne se synchronisent pas (et ne se synchroniseront donc pas plus tard)
- Traiter les observations d'une fenêtre sûre
  - Diagnostic global : ensemble de trajectoires expliquant les observations
    - L'opérateur peut les visualiser et prendre sa décision
- Comment décider d'une fenêtre sûre :
  - Utiliser les connaissances sur les délais bornés dans les canaux + délais entre deux réactions

# *Fenêtre sûre*

- Et si pas possible ?
  - On étend les diagnostics locaux en supposant k observables (par exemple k = taille max du canal)
  - Le diagnostic global calculé sur la fenêtre est optimiste (candidats possibles sous réserve d'observations attendues)
  - Les états finaux/initiaux des diagnostiqueurs locaux sont mis à jour d'après les observations vraiment reçues
  - Au début de la fenêtre suivante, les k premières observations par composant permettent de valider ou non les diagnostics candidats a posteriori
  - Dernière fenêtre (ou fenêtre sûre), on a k=0

# *Conclusion*

- Approche décentralisée du diagnostic
  - Ou distribuée, répartie …
  - Indispensable pour des systèmes complexes
  - Bien adaptée à une vision modèle (composition)
  - Adaptée aux reconfigurations (topologie)
  - Liens avec les approches décentralisées du contrôle :
    - Su-Wonham, Rudie-Wonham, Ricker, van Schuppen …

# FIN

- Pas fait :
  - Autres modélisations :
    - Automates « étendus » : temporels, communicants
    - Réseaux de Petri
    - Algèbre de process
  - Diagnosabilité :
    - Approche composition automates (Huang-Jiang-Chandra-Kumar01)
    - Approche structurelle (Travé-Escobet-Milne01)
  - Représentation économique à base de BDDs
  - Utilisation des techniques de model-checking (accessibilité)