"Diagnostic, diagnosticabilité et réparabilité"

Marie-Odile Cordier, Yannick Pencolé, Thierry Vidal
+ Louise Travé-Massuyès

IAF’08

Partie 1 : Introduction au diagnostic des systèmes à événements discrets
Partie 2 : Diagnosticabilité des SED
Partie 3 : Approche décentralisée / distribuée des SED
Partie 4 : Diagnostic et réparation

Context : Model-based diagnosis

From static systems to dynamic systems …

- Diagnosing static systems :
  - Foundational work by Reiter, de Kleer (1989)
    - Consistency-based approach : conflicts, hitting set algorithm, ATMS, GDE
  - Important contributions of the DX community :
    - Logical formalization
    - Efficient algorithms
    - Abductive approach : causal graph, explicative diagnosis

- Monitoring dynamic systems :
  - Introduction of the temporal dimension :
    - Temporal causal graphs
    - QSIM (Kuipers et al.)
  - Bridge with the control theory community around DES
    - DES : Sampath et al (94) : "…."

Diagnosis models for dynamical systems

- Predictive models (or behavioral models) : able to simulate the system behavior

- Explicative models : describing the (causal/influence) links between the system behavior (faults) and the observation (symptoms) :
  - Temporal causal graphs / influence graphs

- Associative models : from symptoms to faults
  - Expert systems, production rules + time (G2, Chronos …)
  - Chronicle recognition

Predictive models

- Models / continous systems
  - Continous variables (values /continuous time)
    - Differential equations
    - ✓ Numerical models : Control theory
    - ✓ Qualitative models (or semi-quantitative models) : QSIM, Mimic

- Models / discrete-event systems
  - Discrete variables + discrete time

- Hybrid systems :
  - System modes described by differential equations + discrete change of modes

Discrete-event systems

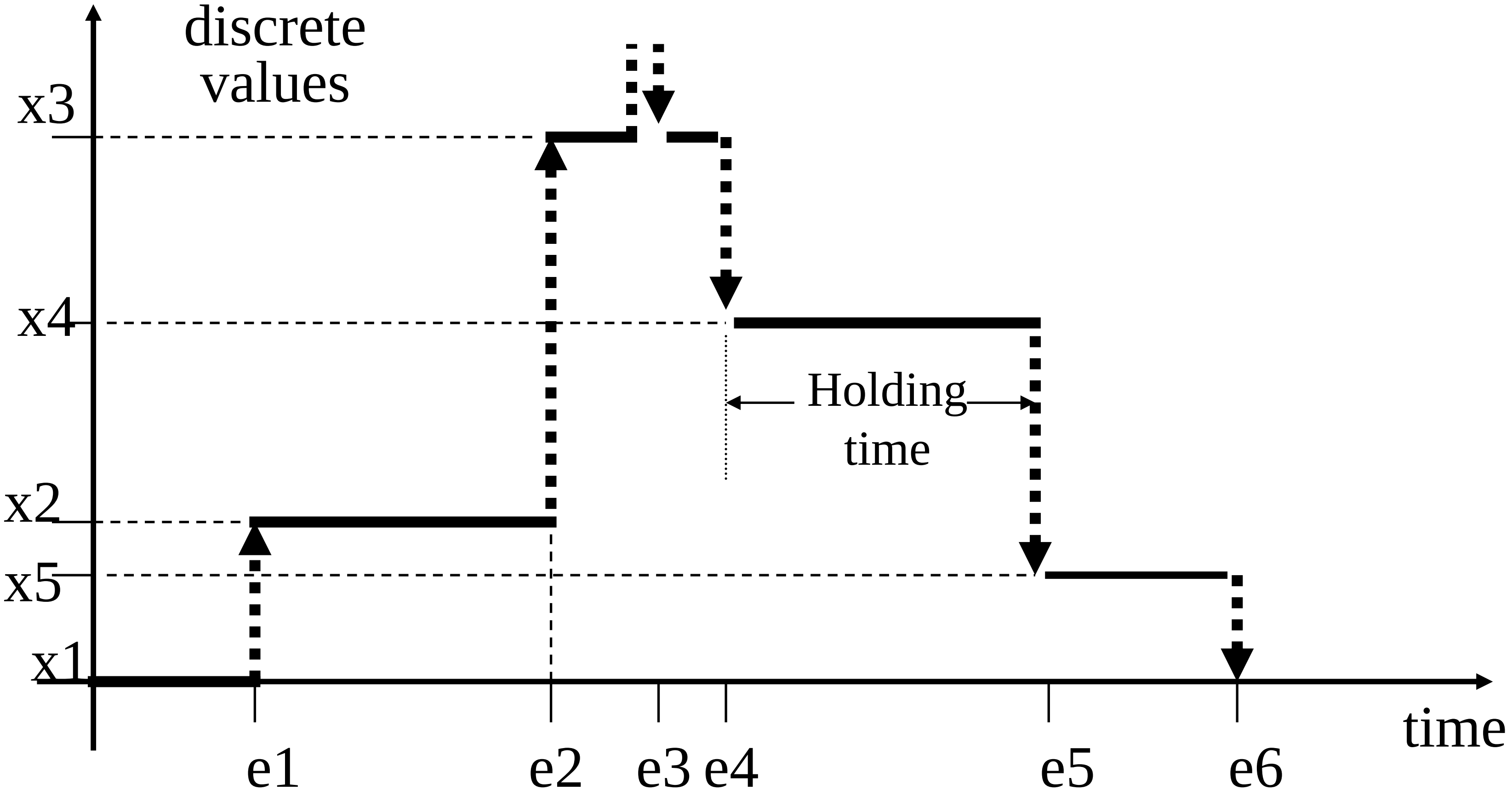
Time-driven vs Event-driven

- Time-driven dynamics (change of state)
  - The system state is considered at each clock tick
  - « time-driven » model
  - Clock Synchronisation

- Event-driven dynamics
  - The occurrence of an event triggers the change of system state
  - « event-driven » model
  - Asynchronous (except for shared events)

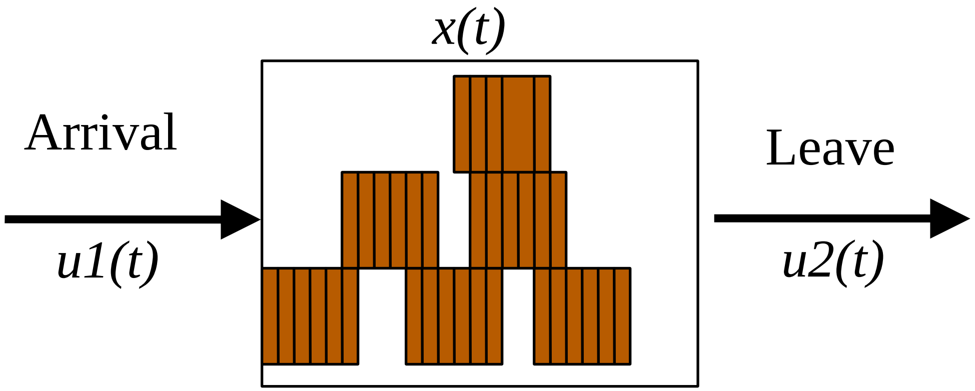
Discrete-event system (DES)

DES : event-driven system (« reactive») + discrete variables



Discrete system (by nature)

- **Warehouse**



  - *x(t)* number of boxes in the warehouse at *t*
  - *u1(t)* = 1 if a box arrives at time t, else 0
  - *u2(t)* =1 if a box leaves at time t, else 0
  - At a given time t, either u1(t) or u2(t)

- **Telecommunication networks** : exchange of messages

System discretization

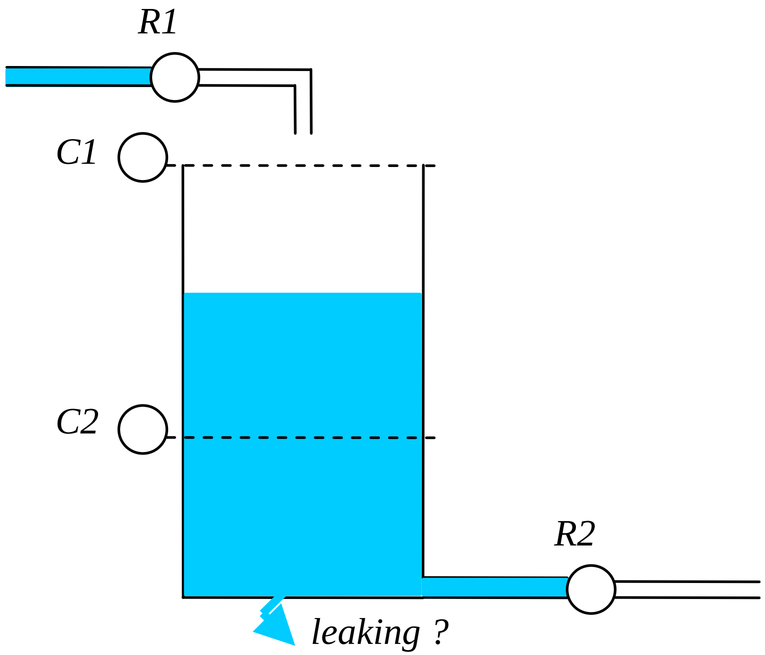
- Continuous system but discrete model
  - the model type is chosen according to the task
  - « Quantization »

Example :

- container

System discretization

- Container



R1 : {open, closed}
R2 : {open,closed}
C1 : {below/above maximal level?}
C2 : {below/above minimal level?}

Discrete state space :
R1 x R2 x C1 x C2

Examples of events :

opening de R1
closing de R2
The water level is below C2
The container is flowing !!

- Fault detection :
  - *R1=closed, R2=closed, C2= below the minimal level*
  - *Alarming event : « the container level is below C2 »*

Discrete-event systems / models

Formalisms for DES

- A DES can be represented as a language
  - Language
    - alphabet : set of events *E*
    - word : sequence of events
  - Behavior of a DES
    - Set of words built on *E*
    - They represent the set of possible behaviors of the system
    - « prefix-closed » language
- Formalisms
  - Algebra
    - Process algebra
  - Transition systems
    - automata, Petri nets

Automata (finite state machine)

- Tool suited to represent a language
- Definition : G = (X, E, f, x0, Xm)
  - *X* : set of states
  - *E* : set of events (alphabet)
  - *f* : transition function $X \times E \rightarrow 2^X$
  - x_0 : initial state
  - X_m : set of final states

« *prefix-closed* » language: all the states of the automaton are final
« *liveness* » : always at least one transition going out from any state

Building the model from the components

- System : a set of components
  - Component models (local model)
    - Library of component models
  - Compose the local models to get the system model (global model)
  - Use of **composition operation** on the component models
- For automata, two basic operations:
  - [Product of automata]
  - Parallel/Synchronous composition

Parallel (synchronous) composition of automata

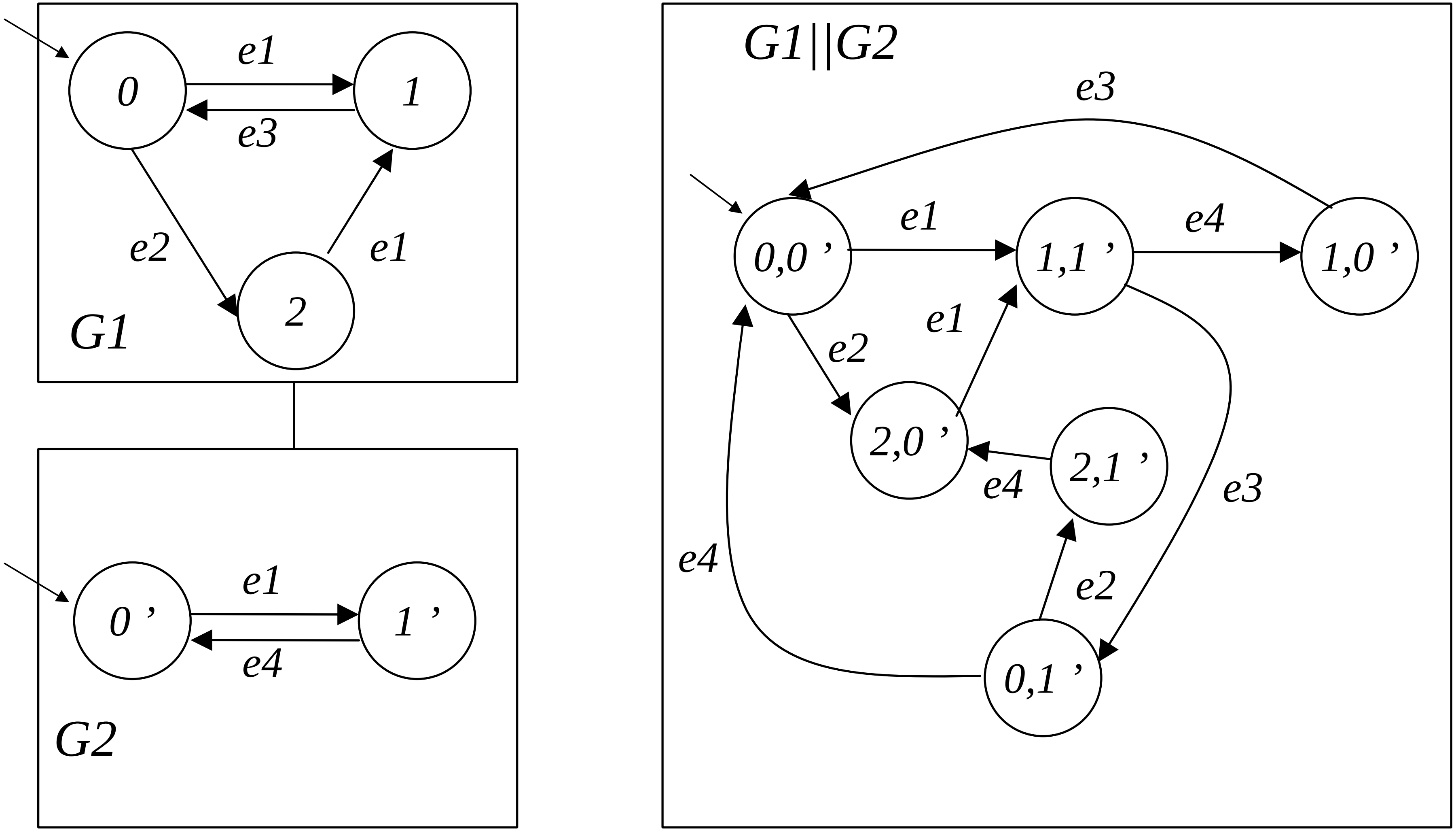
- $G = G1 \parallel G2$
- $G = Acc(X1 \times X2, E1 \cup E2, f, (x_{01}, x_{02}), X_{m1} \times X_{m2})$
  avec

  $f((x1,x2),e) = (f1(x1,e), f2(x2,e))$ *si* $e \in \Gamma 1(x1) \cap \Gamma 2(x2)$
  $f((x1,x2),e) = (f1(x1,e), x2)$ *si* $e \in \Gamma 1(x1) \setminus E2$
  $f((x1,x2),e) = (x1, f2(x2,e))$ *si* $e \in \Gamma 2(x2) \setminus E1$
  else undefined

- Trim operation (or accessibility function) : discard the unreachable states
- When no shared events, synchronisation = cartesian product

The model of two interacting components is built by synchronous composition on shared events

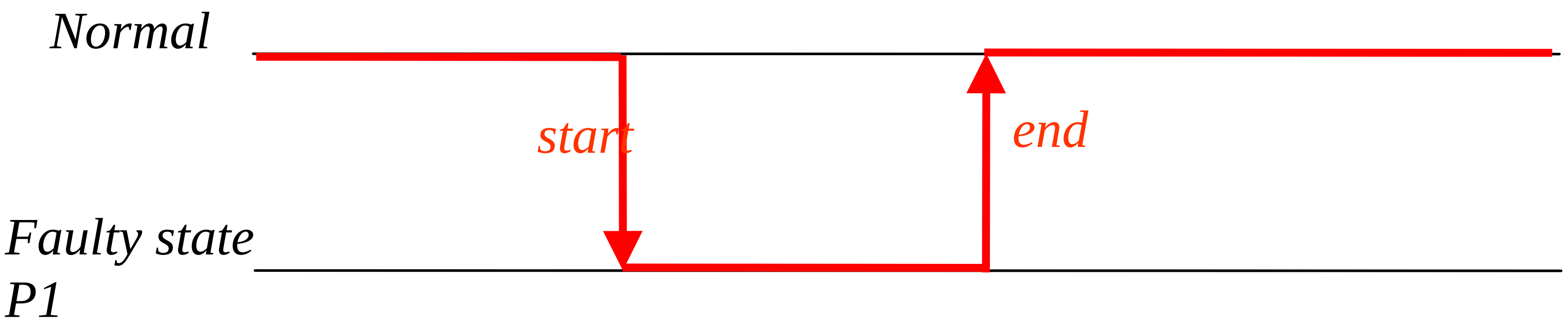
Example


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

Diagnosis of discrete-event systems

Discrete-event systems diagnosis

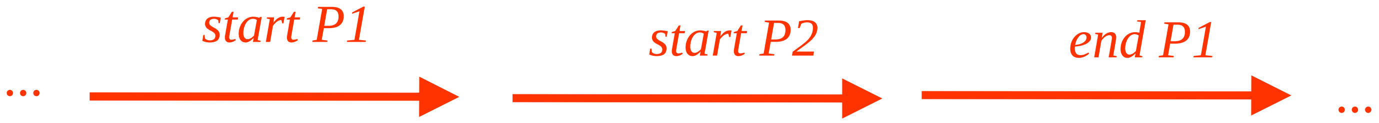
- Faults:
  - Unobservable events
    - Occurrence of the fault (beginning of the fault)
    - End of the fault (intermittent faults)



DES Diagnosis

- Observations :
  - *Observable events*
    - Get through system sensors
    - alarms, notifications, commands…

- Diagnosis :
  - Sequence of events « explaining » what is observed
    - traces, trajectories, histories ...
    - ➢ sequences or sets of faults « explaining » the observations



(DES) Diagnosis problem

- Given a DES model (including fault models)
- Given observations OBS
  - ✓ Sequence / Flow of observed events :
    - ➢ time-stamped observations

➢ Which events can explain what is observed ?

- ➢ Fault events + partial/total order

Diagnosis $\Delta(O)$ = global model $||_{obs}$ OBS

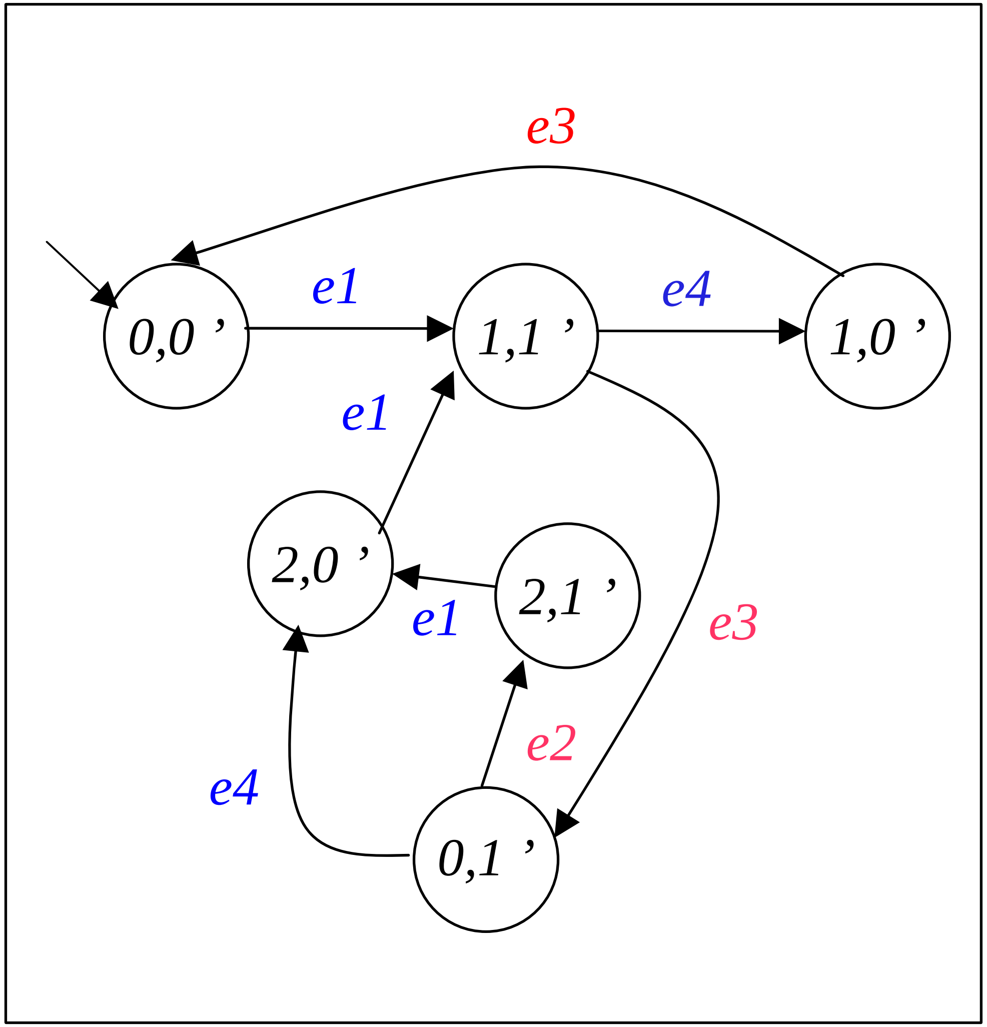
synchronization on observable events obs

On-line vs off-line diagnosis

- Off-line diagnosis
  - All the observations are known from the beginning
  - No real-time constraints

- On-line diagnosis
  - The observations are acquired along time (incrementally acquired)
    - monitoring, surveillance
  - The diagnosis may change (improve) along time when new observation arrive
  - Real-time constraints

Example



Faults events :

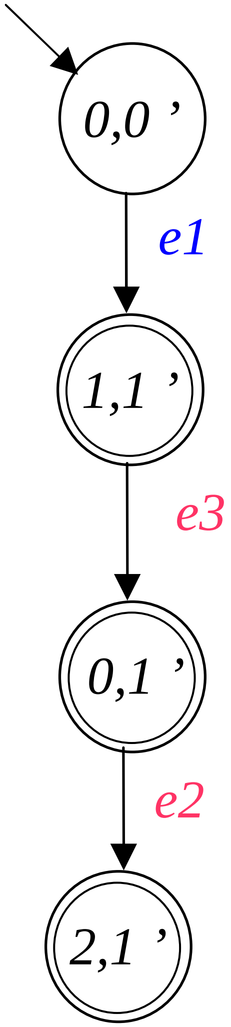
e2, e3

Observable events :

e1, e4

What is the diagnosis
if the initial state is (0,0')
and we observe
e1 followed by *e4* ?

After observing e1

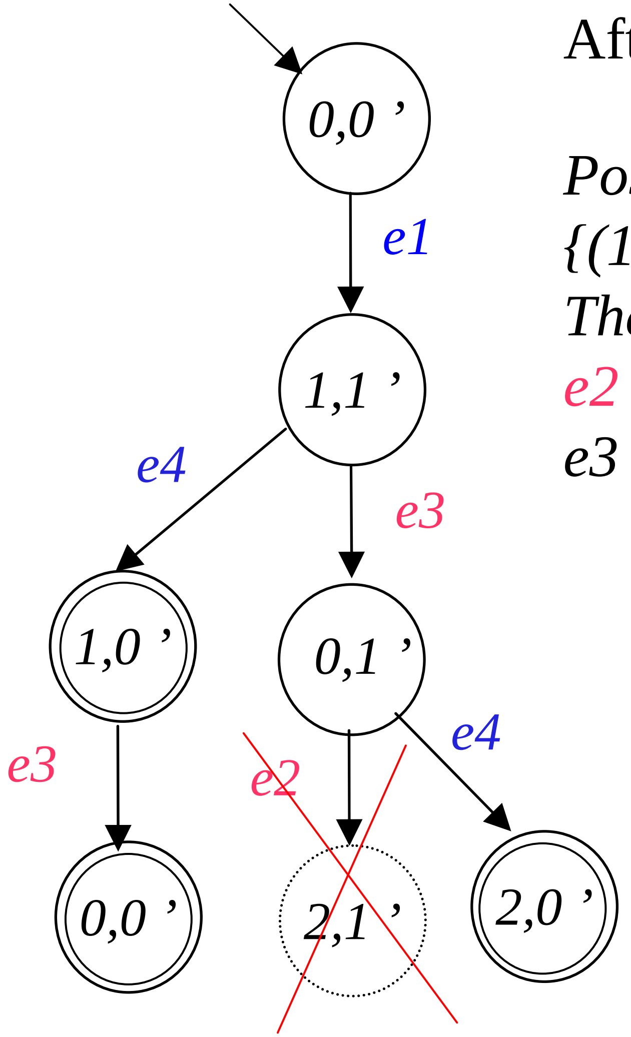


After observing *e1 :*

1) The system can be in one of the 3 states : {(1,1'), (0,1'), (2,1')}

2) e2 and e3 may have occurred.

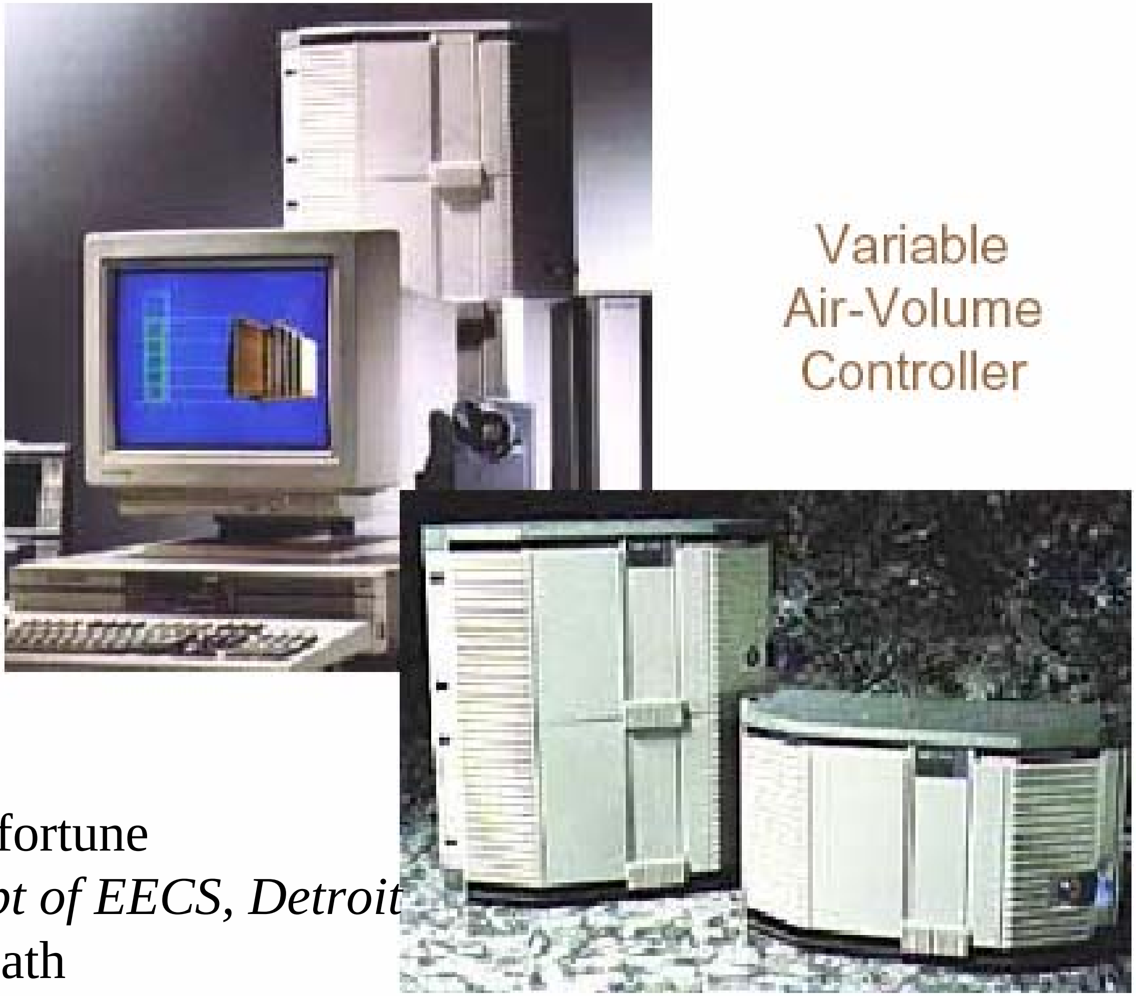
After observing e1 followed by e4



After observing *e1* and then *e4* :

Possible system states :
{(1,0'), (0,0'), (2,0')}
The system may be normal.
e2 did not occur.
e3 may have occurred.

Example : air-conditioned system



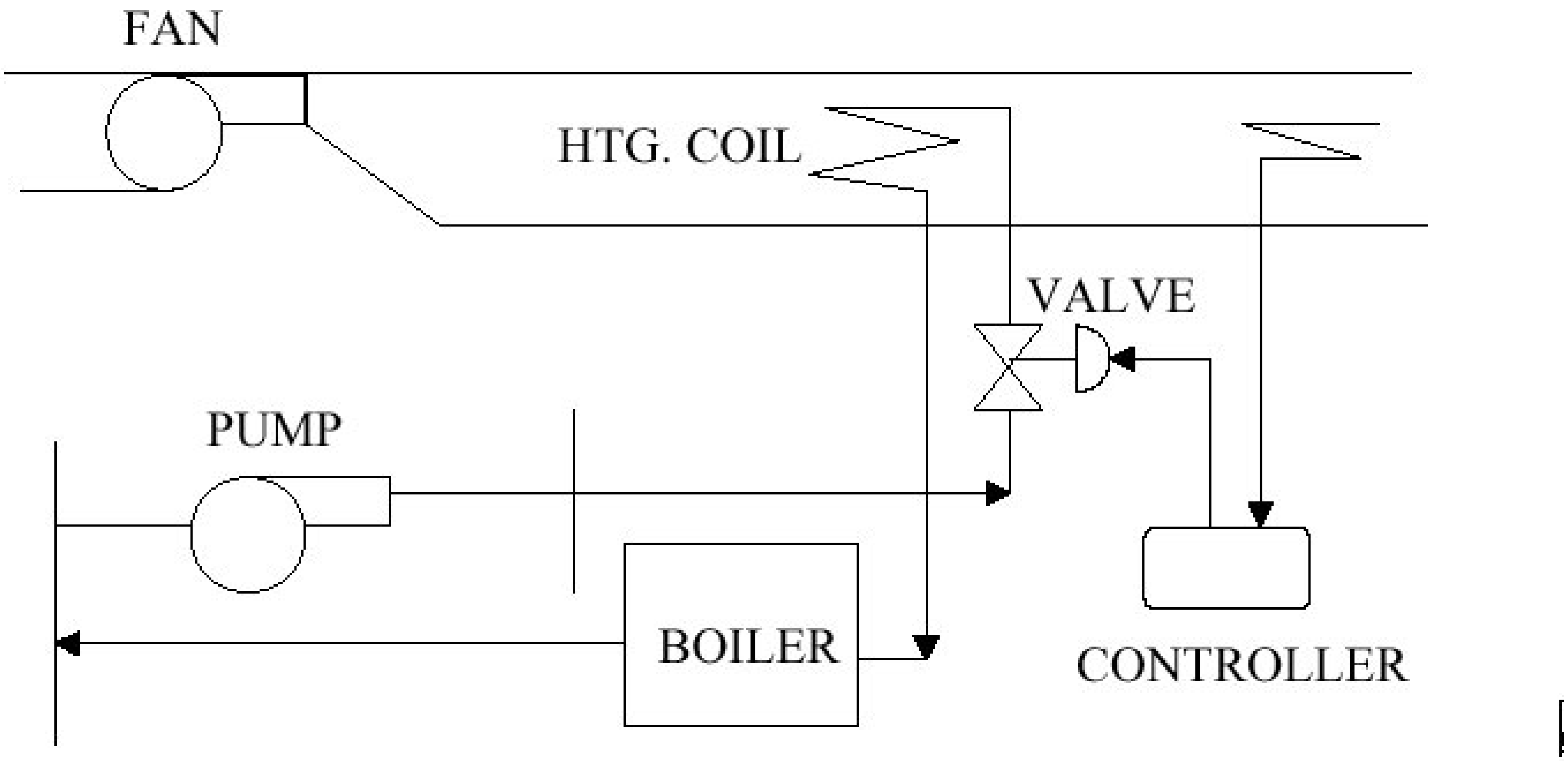
Stéphane Lafortune

Dept of EECS, Detroit

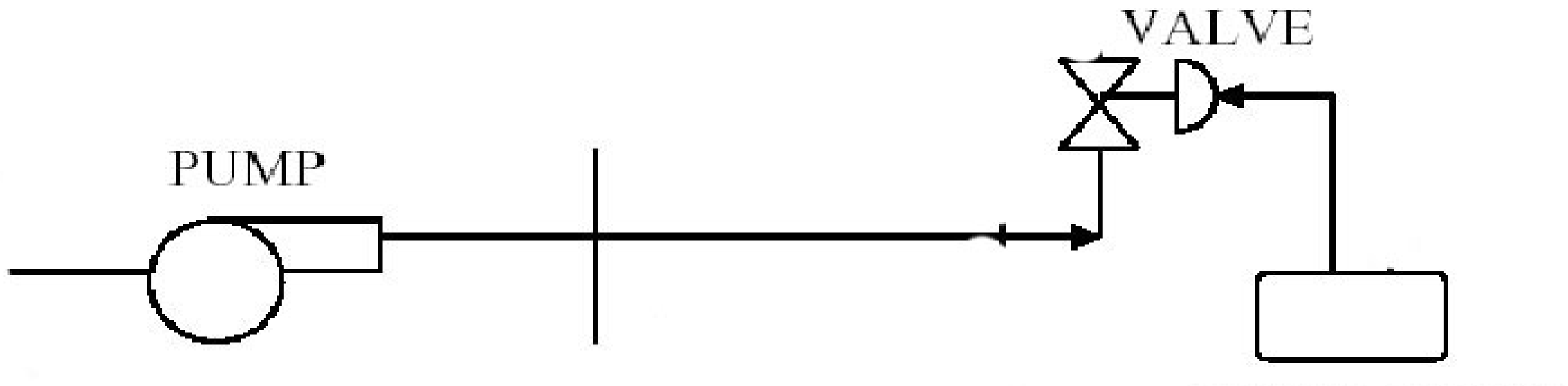
Meera Sampath

Wilson Center for Research & Technology

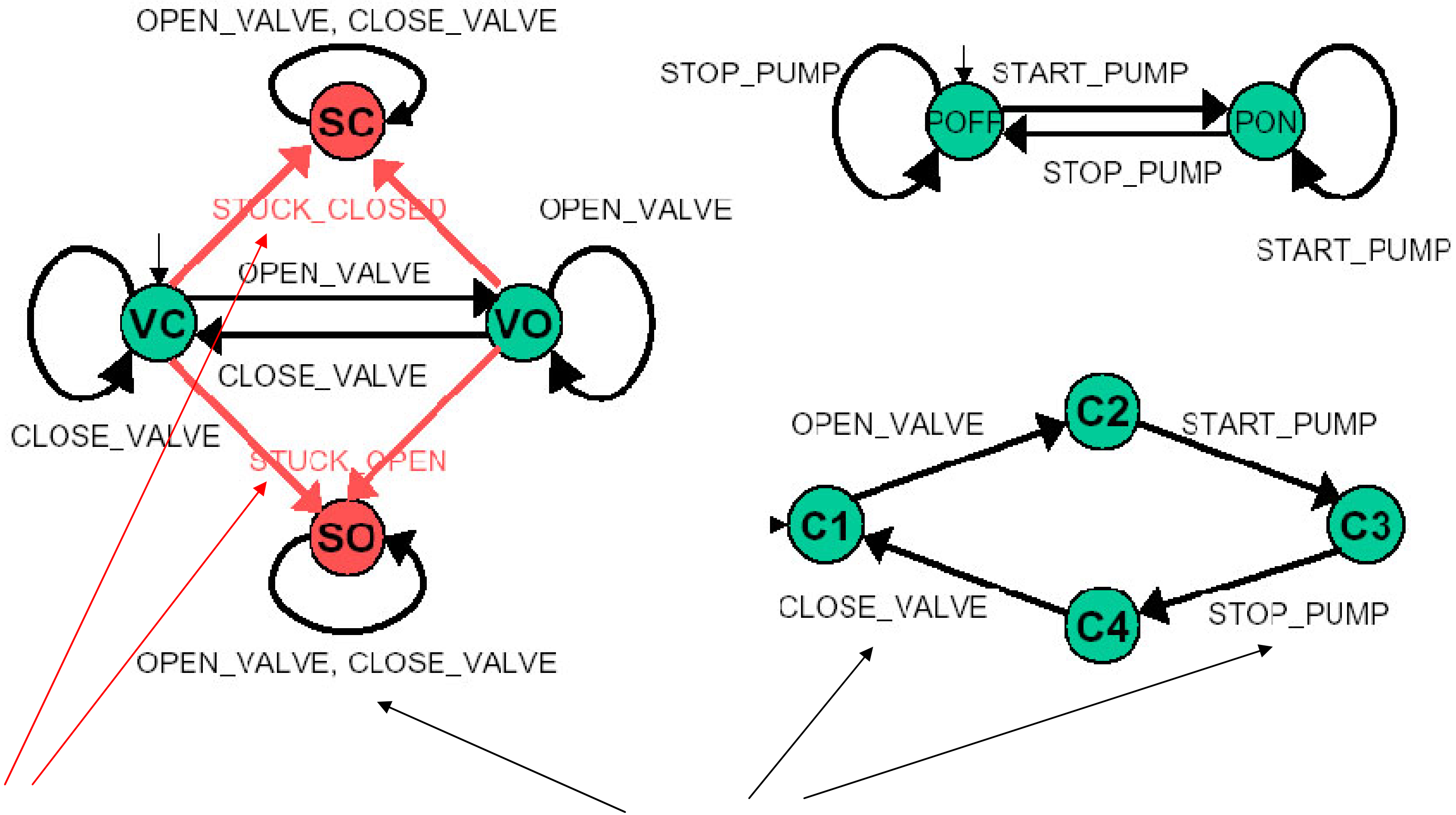
HVAC system



A subpart of the system



Behavioral models of the components



(Permanent) Faults

Control actions

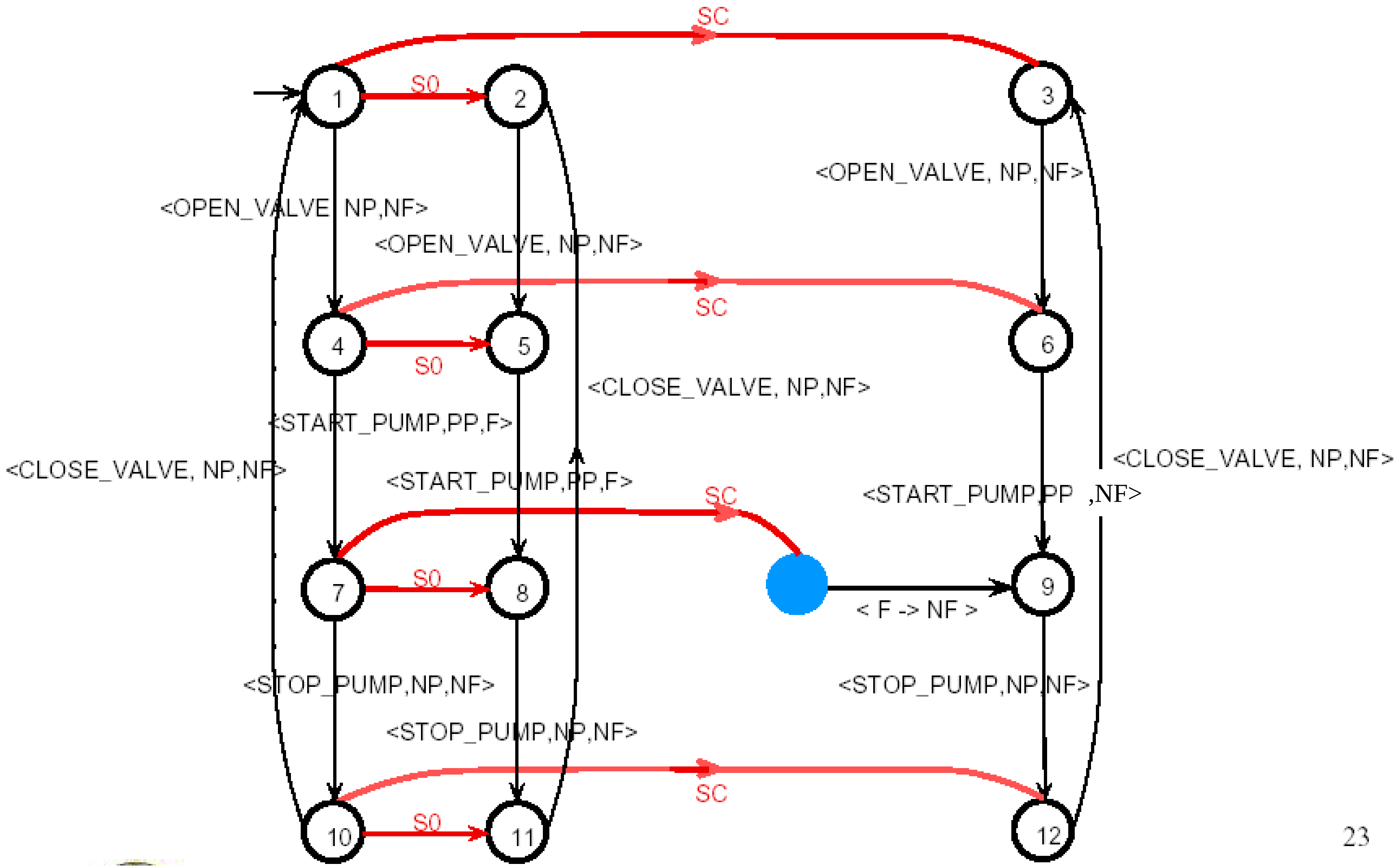
System sensors

- Pressure sensor
  - PP : « Pump Pressure »
  - NP : « No Pressure »

- Air flow sensor
  - F : « Flow »
  - NF : « No Flow »

- Observables : sensor values + control actions

Global model
(got by synchronization operation)



Diagnosis – see automata

Let O be a sequence of observables events:

$$\xrightarrow{OPEN\_VALVE,NP,NF}\quad\xrightarrow{START\_PUMP,PP,F}$$

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

$$\xrightarrow{OPEN\_VALVE,NP,NF}\quad\xrightarrow{START\_PUMP,PP,F}$$

$$\xrightarrow{OPEN\_VALVE,NP,NF}\quad\xrightarrow{SO}\quad\xrightarrow{START\_PUMP,PP,F}$$

$$\xrightarrow{OPEN\_VALVE,NP,NF}\quad\xrightarrow{START\_PUMP,PP,F}\quad\xrightarrow{SC}\quad ....$$

The diagnoser approach

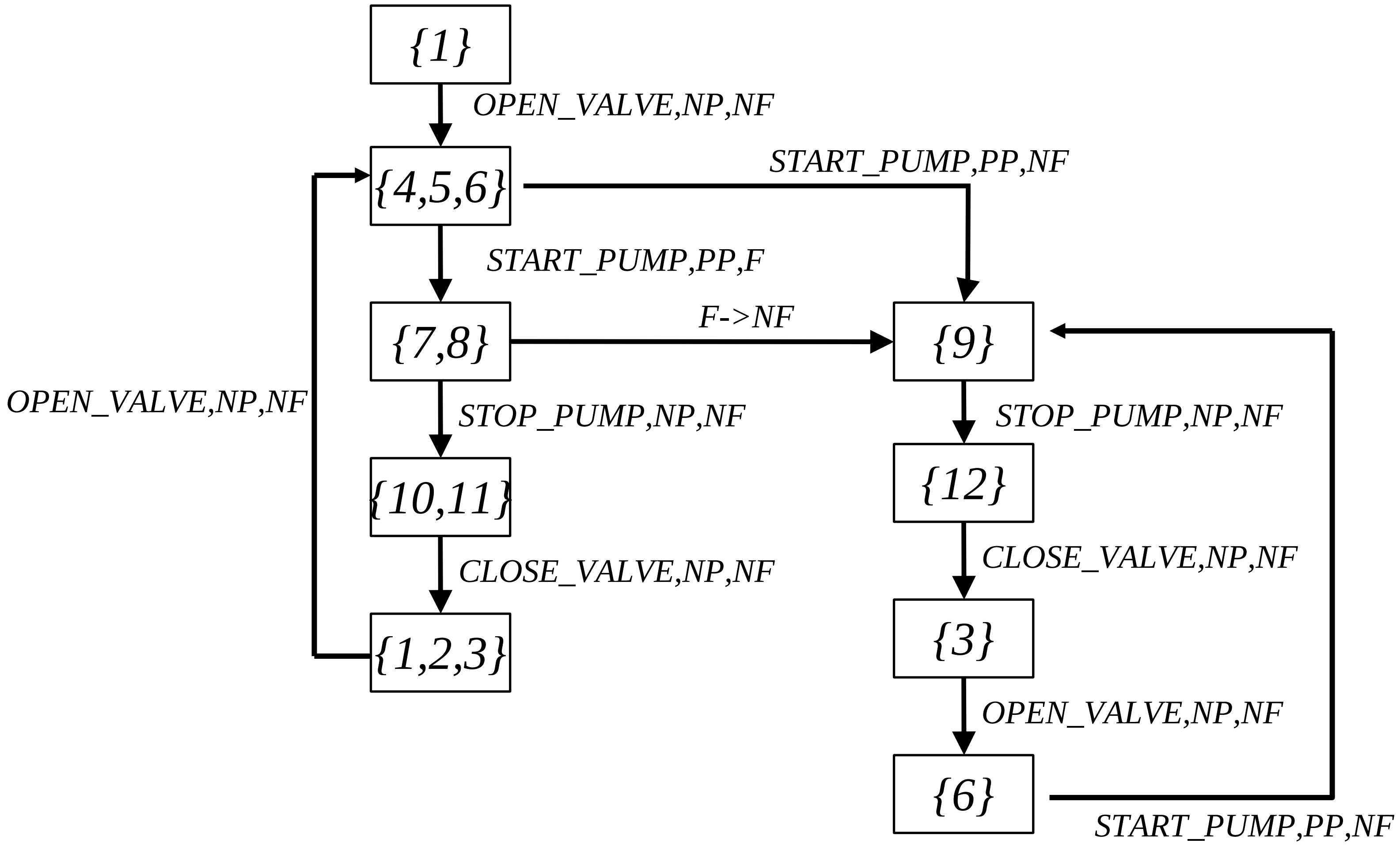
- Simulation-based [Baroni et al.]
  - « unfold » the automaton according to observables
    - complex, adapted to off-line computation

- Diagnoser-based [Sampath et al.]
  - Diagnosis-oriented compilation
    - More efficient when on-line monitoring

Observer

- Observer automaton
  - Describes all the observable behaviors of a system. It is a deterministic automaton with observable transitions.
  - An observer state is a belief state describing the possible states of the system after having observed a set of events
  - Built from the system global model by ε-reduction (ε : unobservable event)

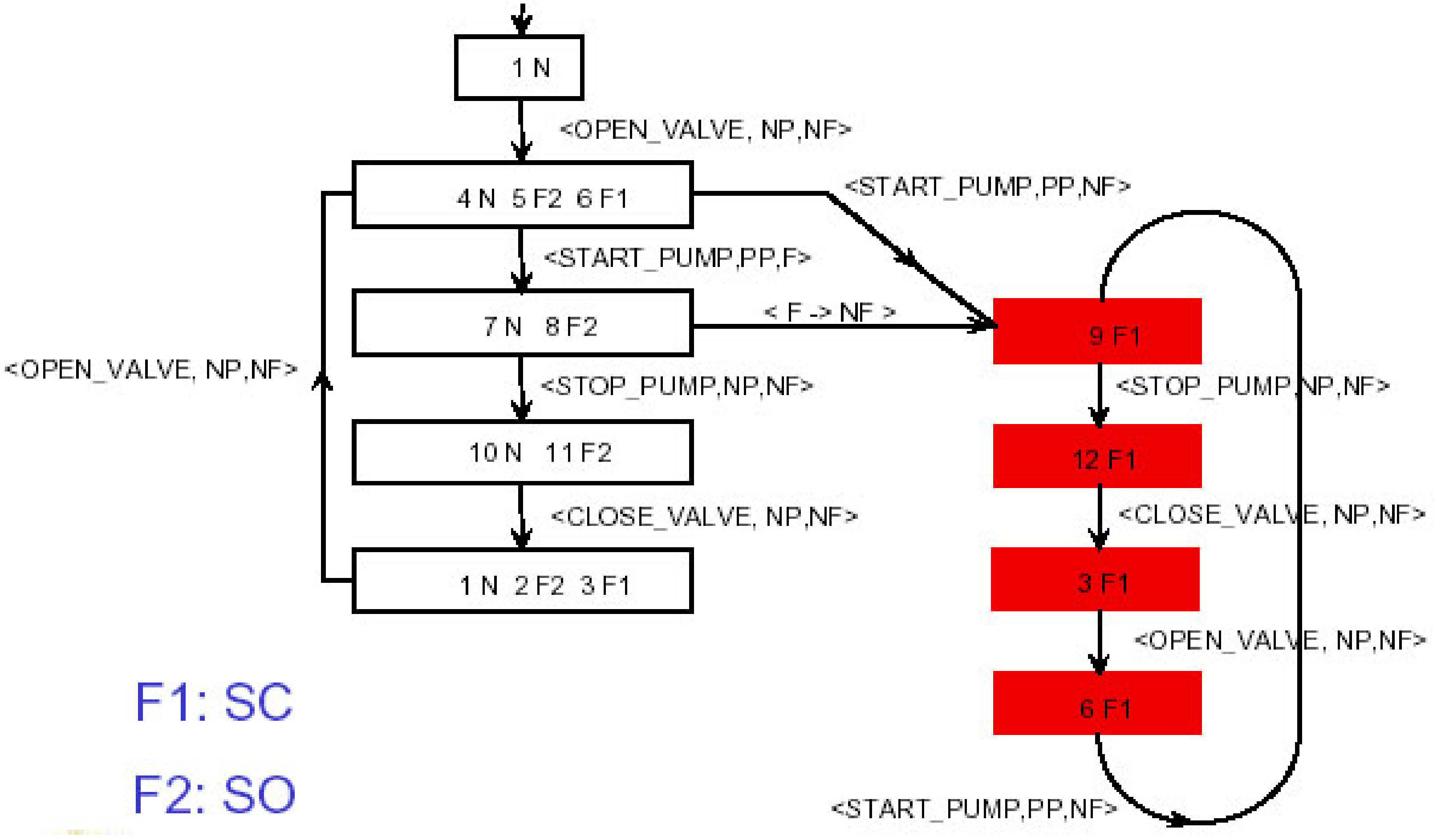
Observer of HVAC



Diagnoser

- **Observer** : to « track » the system thanks to observable events
  - Diagnosis information :
    - The possible states of the system
    - No information on past fault events
- **Diagnoser** [Sampath et al.] :
  - observer + labels
  - The set of occured fault events are stored in the labels

Diagnoser of HVAC



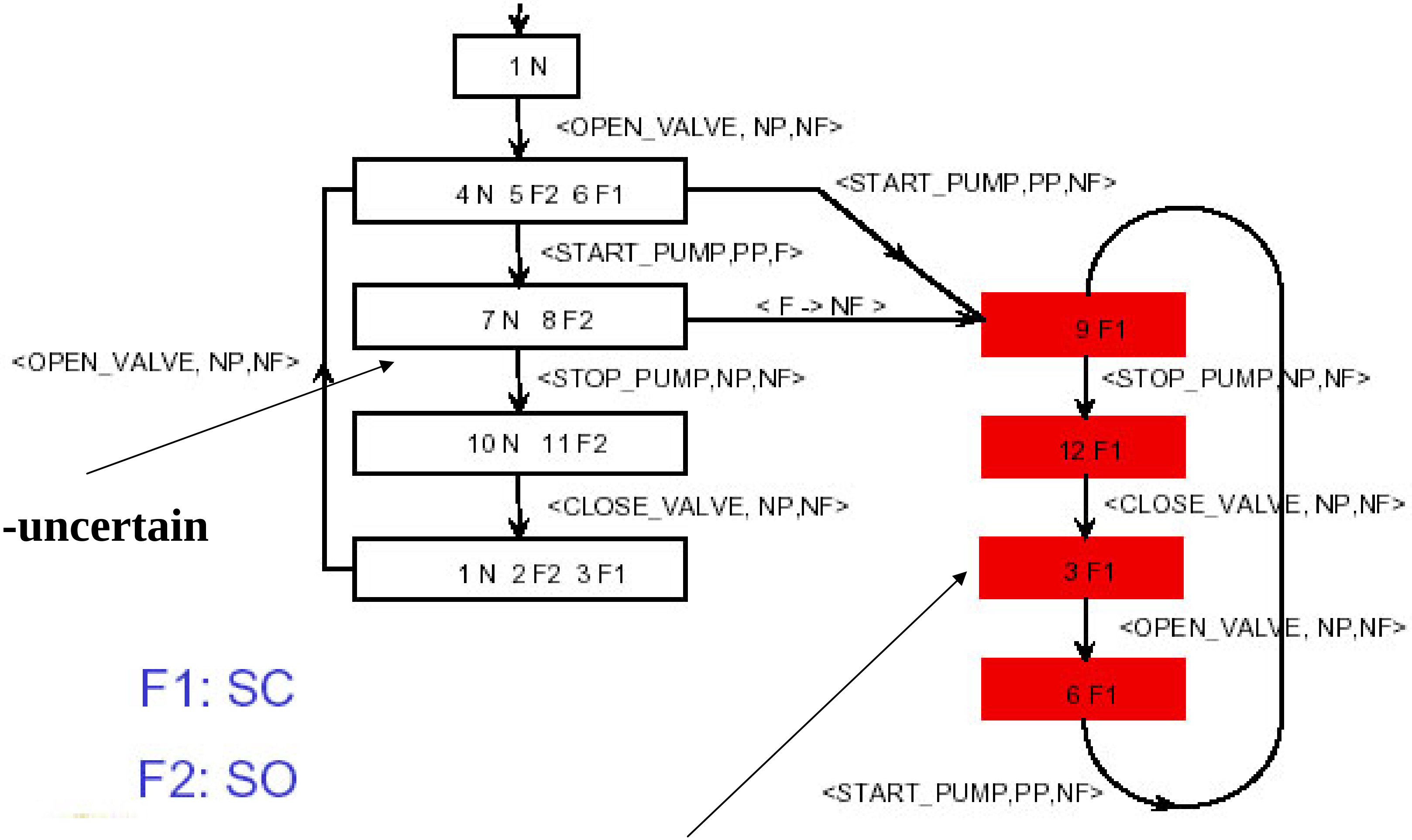
F1: SC

F2: SO

Diagnoser

- Diagnoser states :
  - Set of diagnosis candidates as set of pairs (state, label)
    - Label : fault event on the path arriving in the state, N (normal) when no fault
  - The belief state is said to be :
    - Normal : all the candidates are labeled as N
    - Fi-certain : Fi is in the label of all the candidates
    - Fi-uncertain : else

Diagnoser of HVAC



F2-uncertain

F1: SC

F2: SO

F1-certain

Diagnosis algorithm

- Given what is observed, what are the current diagnosis candidates ?
  - Algorithm:
    - Track the system using the diagnoser according to arriving observation
    - The belief state gives the current diagnosis candidates

Diagnosis algorithm

- Given what is observed, did the fault f occur?
  - Algorithm :
    - Track the system using the diagnoser according to arriving observation
    - The belief state gives the current diagnosis candidates

- Is the algorithm bounded?
  - In case of fault, is it sure that the diagnoser will diagnose the fault in a bounded delay after the occurrence of the fault?

DIAGNOSABILITY!

A first conclusion

- Use of a behavioral model (automata + other formalisms)
  - Normal and faulty behaviors for detection and diagnosis
  - Adding temporal constraints ?
  - Size of the global model? Decentralized/distributed approaches
- Observations
  - Ordered set of observations
  - But what if delays between emission/reception? What is clocks are not synchonized? Take into account partial ordered observations
  - What if uncertain observations? Observations represented as automata
- Diagnoses : trajectories, defined by SD ⊕ OBS
- Algorithms :
  - Unfolding of the automaton and search for the consistent paths
  - Strategy based on preference criteria (probabilities?)
  - Efficiency : Compilation of the automaton into a labelled observer (diagnoser)
  - Compact representation of trajectories? Partial order reduction, BDD, Model-checking tools?
  - Termination? Diagnosability?

Diagnosability of Discrete-Event Systems

Marie-Odile Cordier / IAF’08

Diagnosability

Diagnosability

- Diagnosability:
  - Given the system sensors (and the observable events), decide which faults can be diagnosed for sure and in a bounded delay

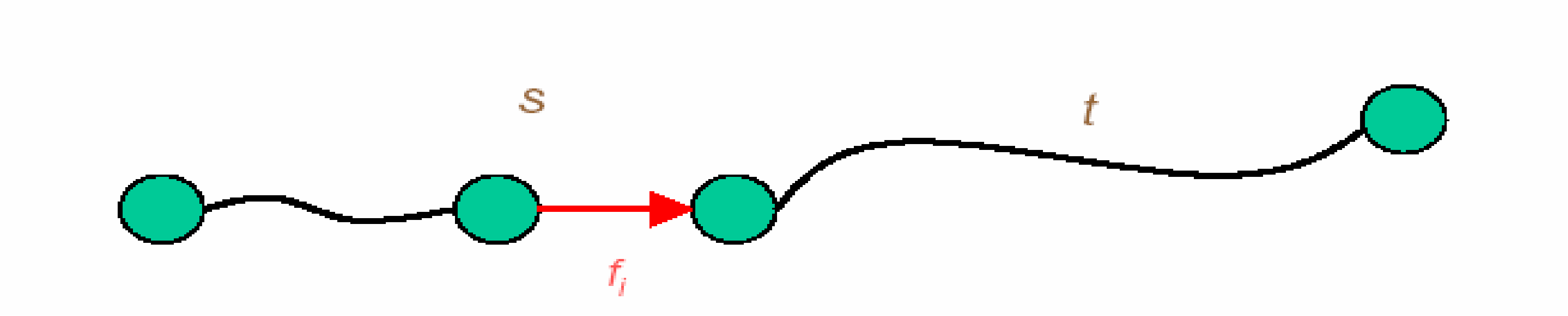
- Goals of the diagnosability analysis
  - At the diagnosis time :
    - Be sure to get the diagnosis in a bounded delay after the occurrence of the fault
    - Or being aware of undiagnosable faults, trigger adapted repair actions, without waiting too long
  - At the design time :
    - Improve the observability of the system (what to observe, where locate the sensors) to get a diagnosable system?
    - Or, being aware of undiagnosable faults, look for adapted repair actions dealing with these uncertain cases to get a self-healable system

Definition by Sampath et al. Unformally …

- Partition Π*f* of fault events into fault types:
  - Each fault event belongs to one fault type
- Sequence O of observations

- A DES is *diagnosable* iff any *occurrence of a fault type of* Π*f* can be diagnosed for sure, in bounded time after its occurrence, from the observations O

Diagnosability



- Path *s* ending by a fault event *fi*
- Path *t* continuing *s*
- Each path « looking similar to » *s*.*t* contains a fault event *fi*
  - « looking similar to » = having the same observable behavior (same observable projection)

Formal definition

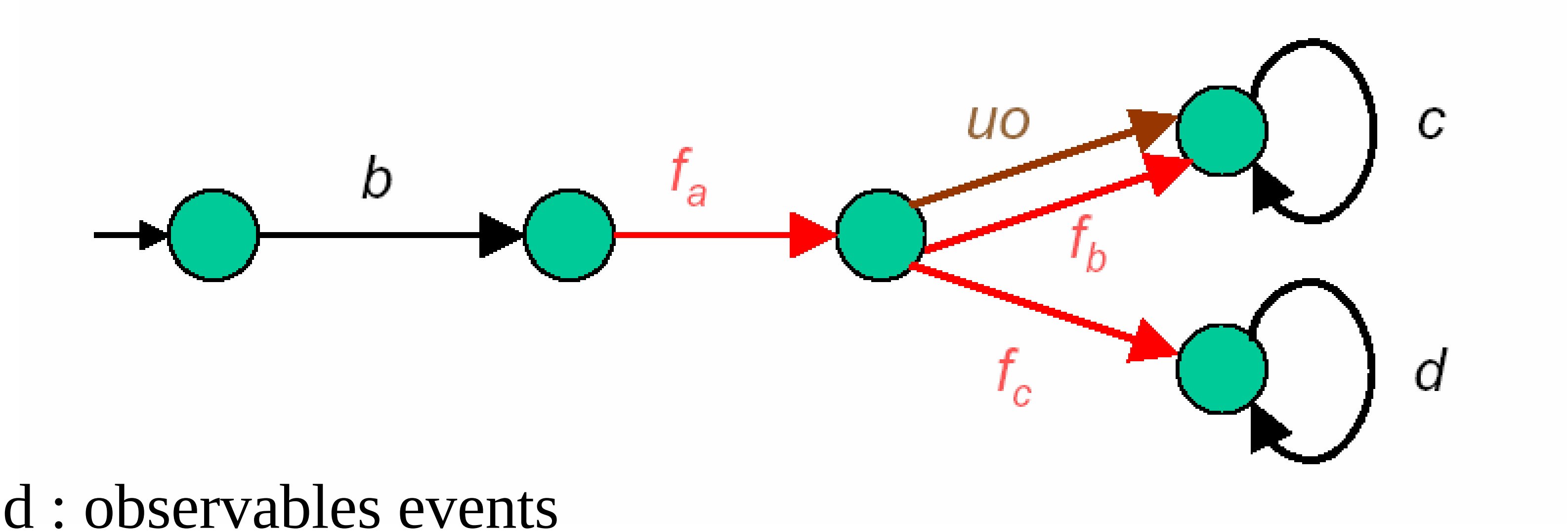
- A prefix-closed and live language L is diagnosable wrt a projection *P* and a partition Π*f* on events *Ef* iff :

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

- D : *diagnosability condition*

$$\omega \in P_L^{-1}\,[P(st)] \ \Rightarrow E_{fi} \in \omega\ .$$

Example



b,c,d : observables events
uo : unobservable events
fa, fb, fc : fault events (unobservables)

If Πf = { {fa}, {fb}, {fc} } then the system is not diagnosable
If Πf = { {fa,fb}, {fc} } then the system is diagnosable

Diagnosability degree

- A system is diagnosable for a partition of faults Πf

can be rewritten as

- The degree of diagnosability of a system is the « finest » partition(s) for which it is diagnosable
  - The worse case is when the partition gathers all the faults : the system is detectable (at best)
- See our work on self-healability (definition of a diagnosability degree … removing the partition condition)

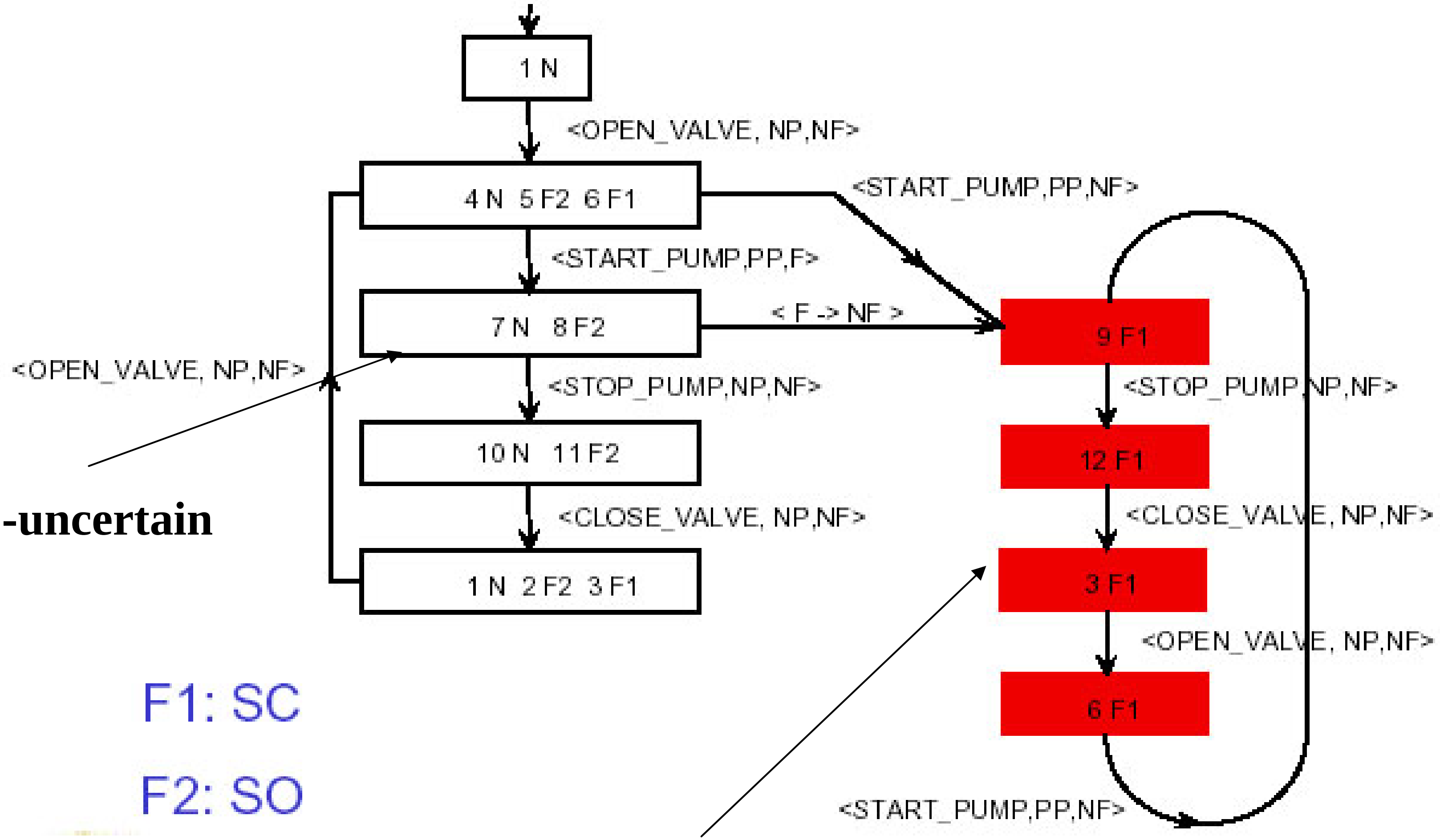
Diagnosability and diagnoser

- Formal result (by Sampath et al. 95) :
  - A DES is diagnosable iff the diagnoser does not contain any uncertain cycles
  - where an uncertain cycle is :
    - A cycle of Fi-uncertain states in the diagnoser

    +

    - The states of this Fi-uncertain cycle must correspond to a (observable) cycle in the system model

Example:
Diagnoser of HVAC



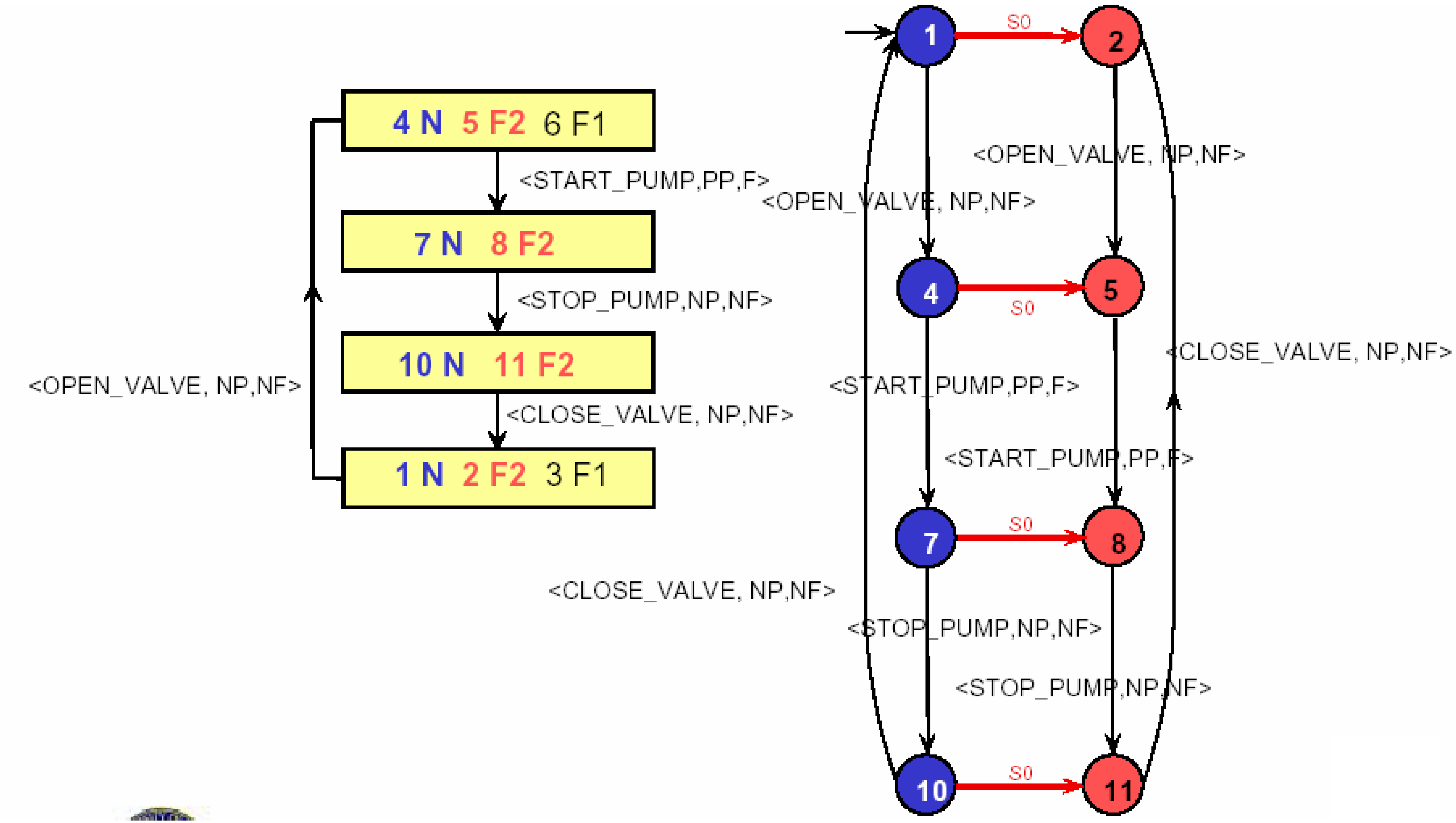
F2-uncertain

F1: SC

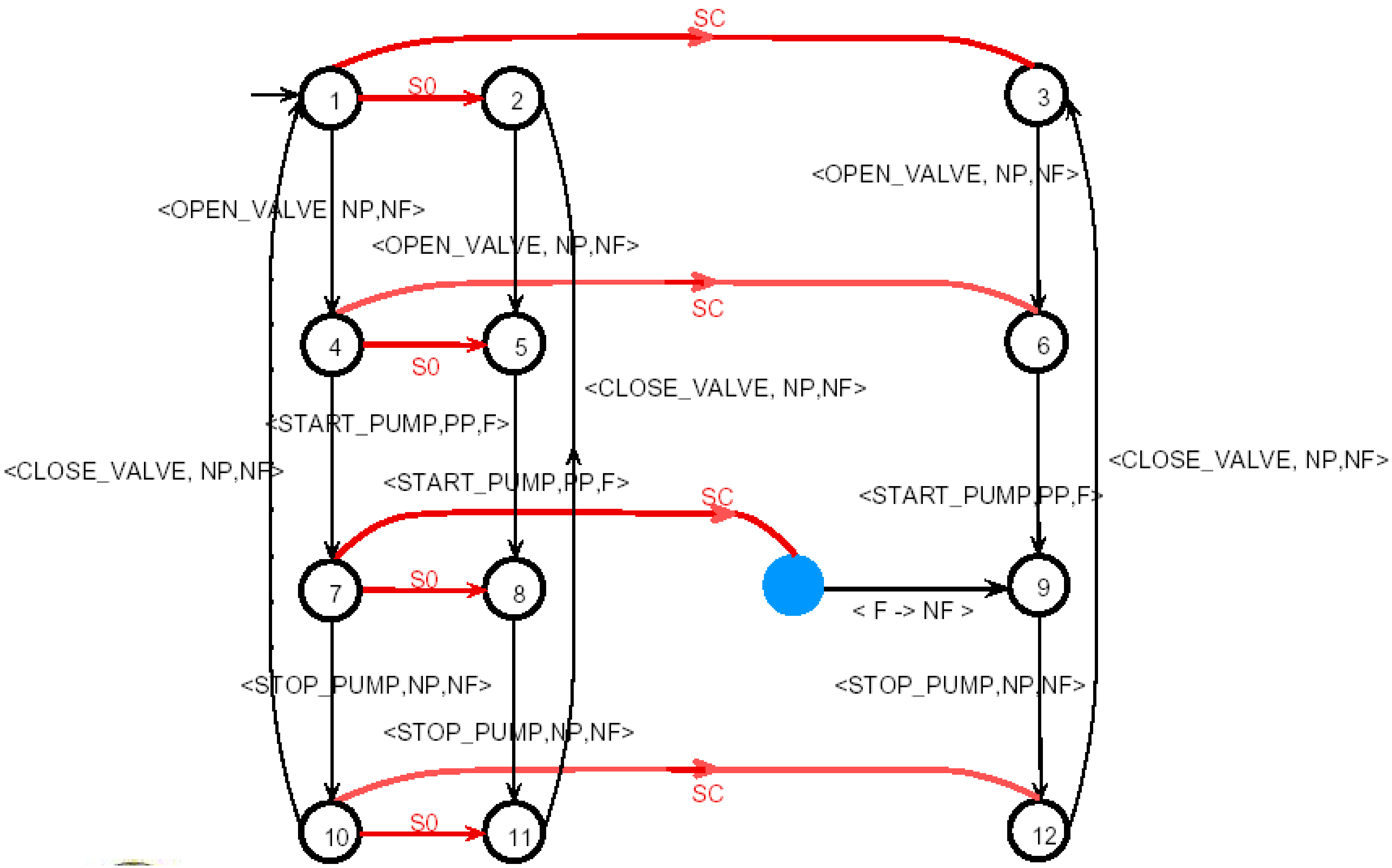
F2: SO

F1-certain

Example

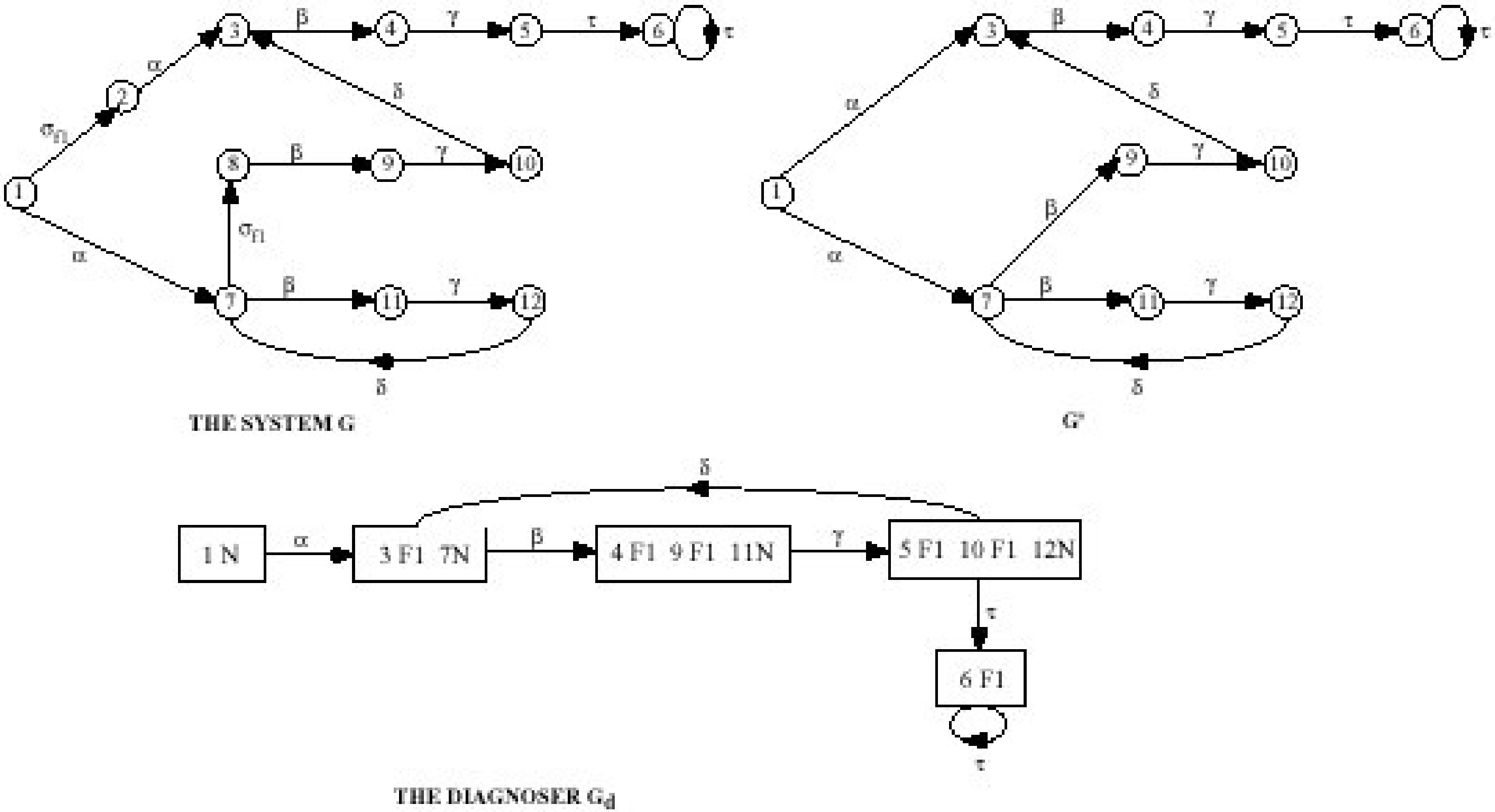


Global model
(got by synchronization operation)



state …

but the system is diagnosable



THE SYSTEM G

G'

THE DIAGNOSER G_d

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

uncertain state … the system is NOT diagnosable

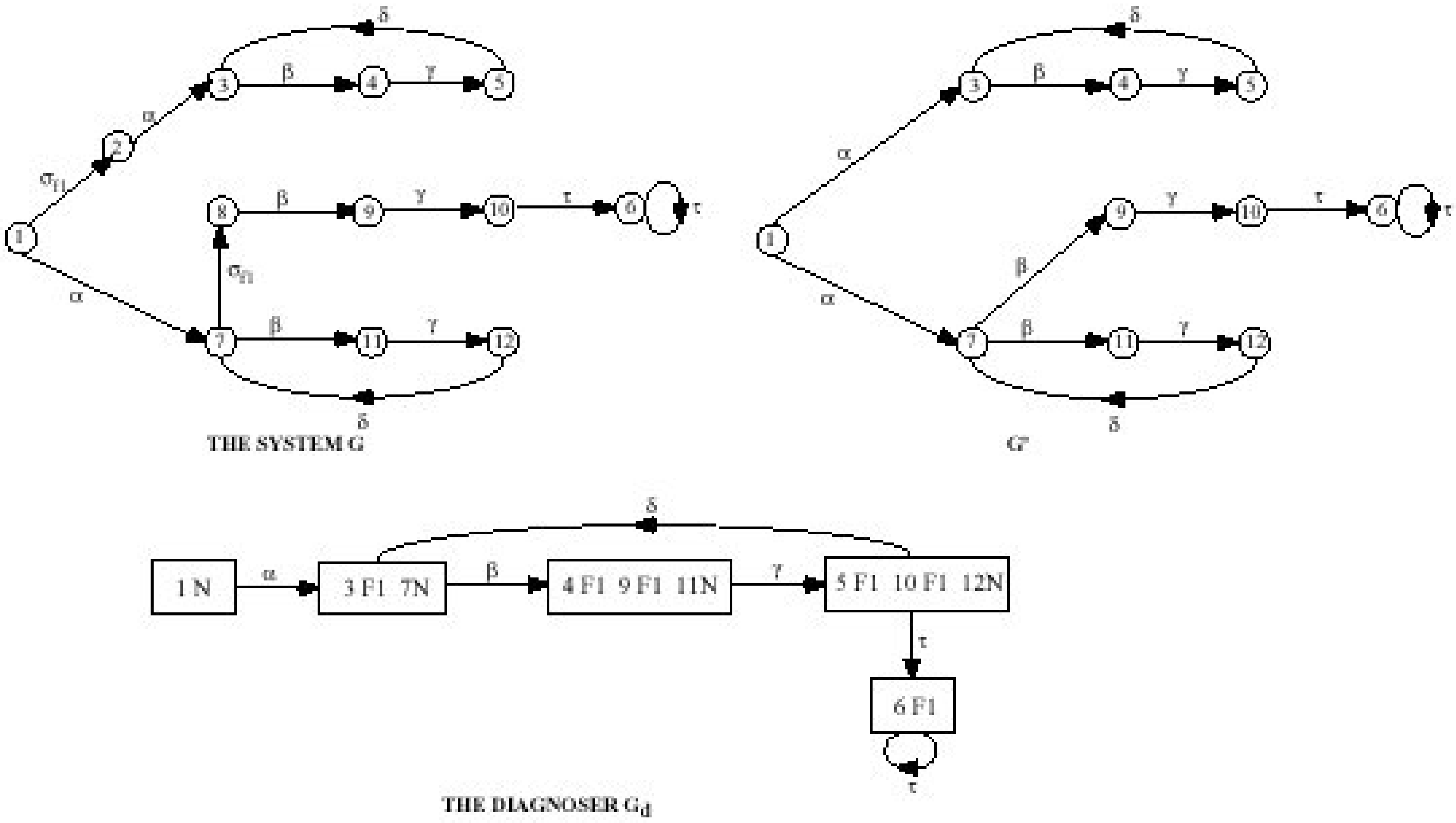


Figure 3.6: Example of a system with an F_1-indeterminate cycle in its diagnoser G_d

analysis using self composition (Jiang et al. 2001)

- Self-product:
  - Build a undeterministic observer G_o after having labelled each state x of the system automaton G by the set of fault events {Fi} that are on the path arriving on x
  - **Compose two copies of G_o**
  - Check that the resulting automaton G_d does not contain any cycles of (belief) states labelled by states with different labels. If no such cycles, the system is diagnosable, else it is not.

Example

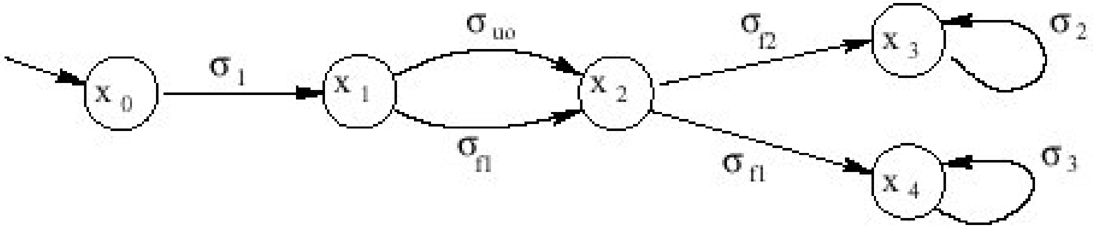


Figure 1: Diagram of the system G

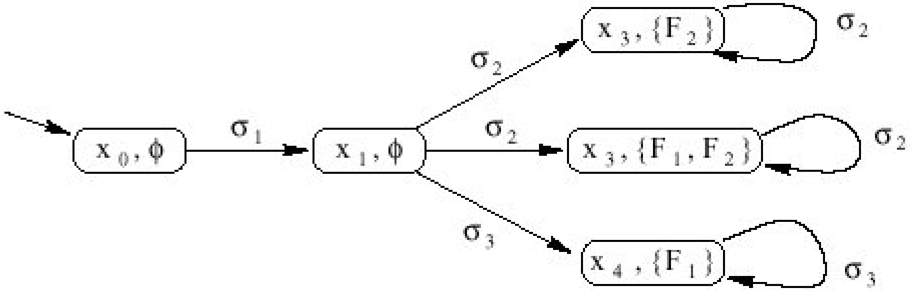


Figure 2: Diagram of G_o

Example

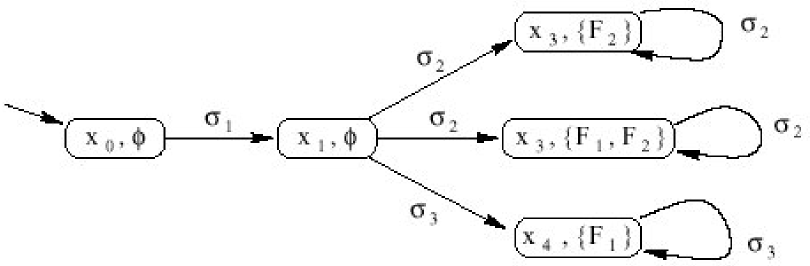


Figure 2: Diagram of G_o

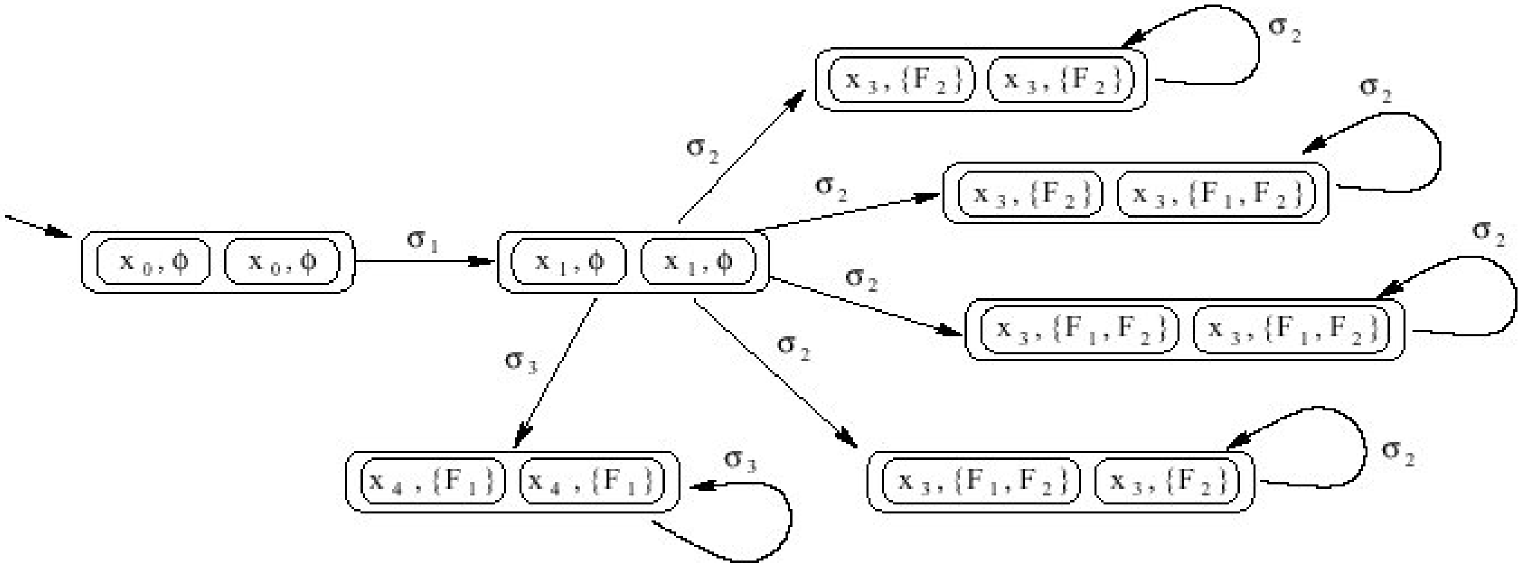

σ_2
$x_3, \{F_2\}$ $x_3, \{F_2\}$
σ_2
σ_2
σ_2
$x_3, \{F_2\}$ $x_3, \{F_1, F_2\}$
σ_2
x_0, ϕ x_0, ϕ
σ_1
x_1, ϕ x_1, ϕ
σ_2
$x_3, \{F_1, F_2\}$ $x_3, \{F_1, F_2\}$
σ_3
σ_2
σ_3
$x_4, \{F_1\}$ $x_4, \{F_1\}$
σ_2
$x_3, \{F_1, F_2\}$ $x_3, \{F_2\}$

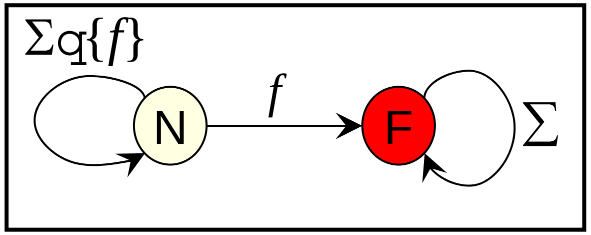
Figure 3: Diagram of G_d

Diagnosis / diagnosability extended to pattern supervision

(T. Jeron, H. Marchand, S. Pinchinat, M.-O. Cordier)

Supervision patterns for the diagnosis (Jeron et al, Wodes'06)

- ❑ A system to be supervised/diagnosed is given by
  - ➢ A prefix-closed model of the system :G = kQ,Σ,q_0, Q, \$ lavec Σ=Σ_oˆΣ_{uo}
  - ➢ A Supervision Pattern : *reachability property*



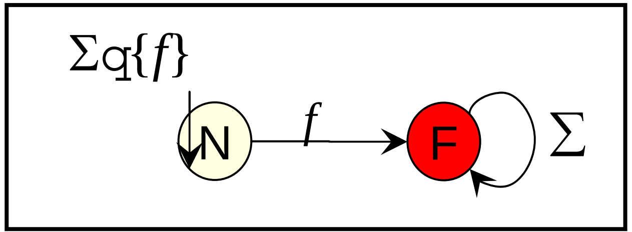
, modeled by an LTS Ω =kQ_Ω,Σ, $q_{0\,\Omega}$,Q_P,\$ $_\Omega$l

✓deterministic, complete

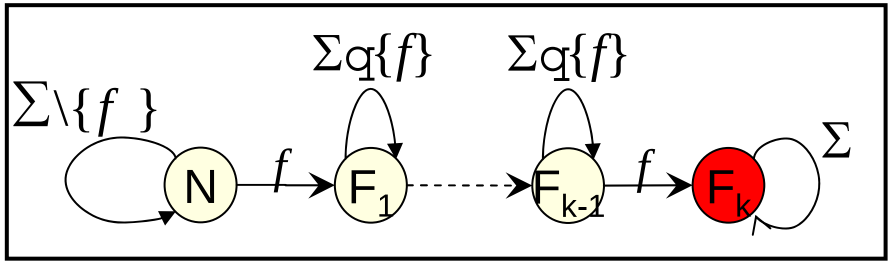
✓with a set of final states Q_P , Q_P » Q_Ω which is stable $L(\Omega,Q_P)=L_{Q_P}(\Omega)$

- ❑ Sequences under supervision : s 5 $L(G)$ _ $L_{Q_P}(\Omega)$

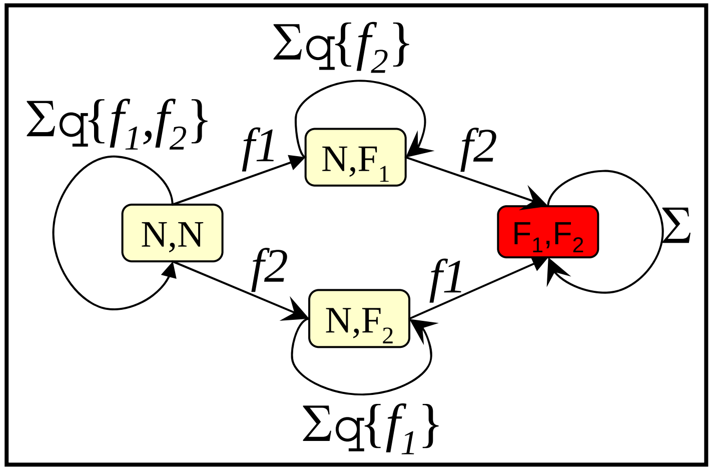
Examples of supervision patterns



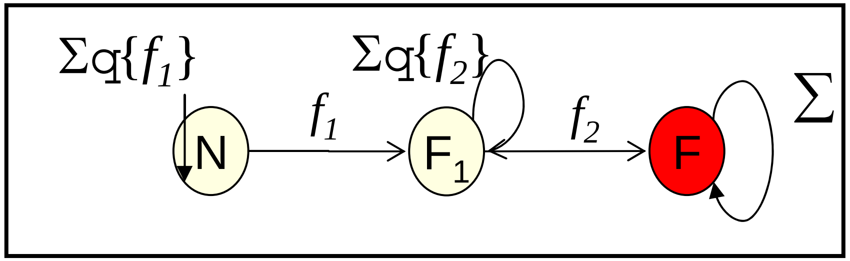
Occurrence of a fault



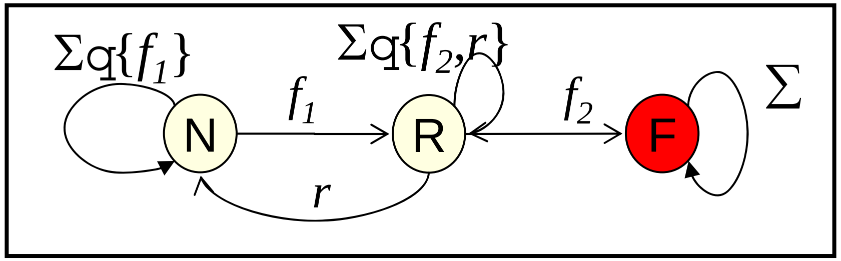
Multiple Occurrence of a fault



Occurrence of 2 faults



Ordered occurrence of events



Intermittent faults with repair

Stability of the final states :
“to have seen f1 at least 2 times” is ok but not “exactly 3 times”

Extended Diagnosis Problem

Problem:

S: known system, partially observable: $\Lambda = \Lambda_{uo}$ ˆ Λ_{o}

Ω : property on executions, L_{Q_P} (Ω) (e.g. occurrence of fault f: L_{Q_P} (Ω) =Σ^{*}.f.Σ^{*})

Does Ω hold on executions [[μ]] compatible with observation μ ?

Construction of a function

Diag(*G*,Ω): *Tr*(*G*) $ {Yes, No, ?} that has to be

(C) Correct : Diag(*G*,Ω)(μ) = No / #[[μ]] _ L_{Q_P}(Ω) = >

= Yes / [[μ]] » L_{Q_P}(Ω)

(B) Bounded: **<*n*, ;*s* 5 [[*μ*]] _ L_{Q_P}(Ω), ;*t* 5 *L*(*G*)/*s* _ $\Sigma^{*}.\Sigma_{o}$,**

mP(*t*)mm¿ *n* , Diag(μ.P(*t*))=Yes

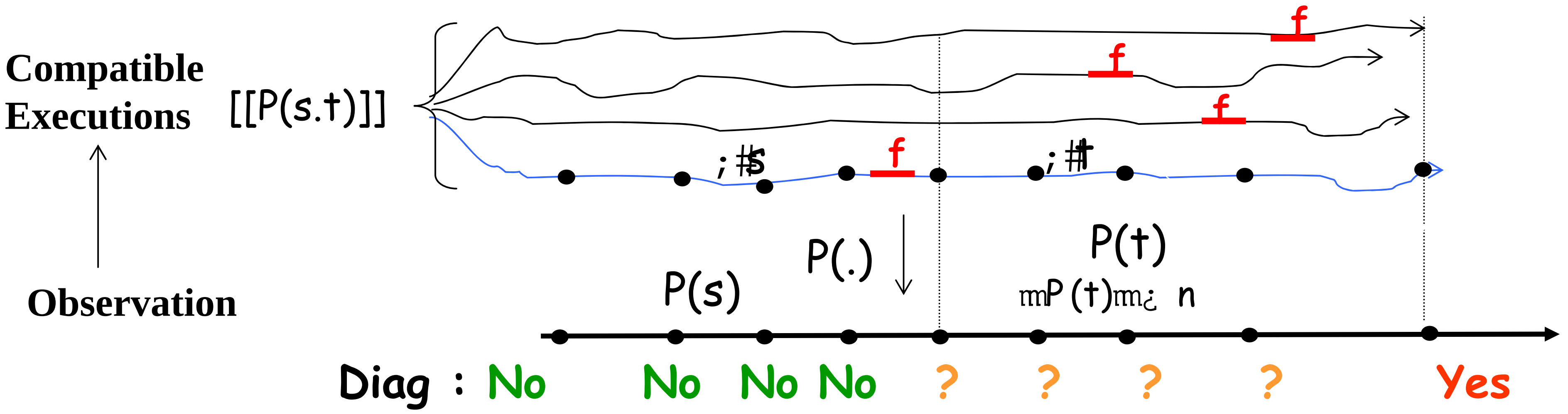
Construction of a function **Diag(*G*,Ω)**: *Tr*(*G*) ! {Yes, No, ?} that has to be

((C) Correct : Diag(*G*,Ω)(μ) = No / #[[μ]] _ L_{Q_P}(Ω) = >

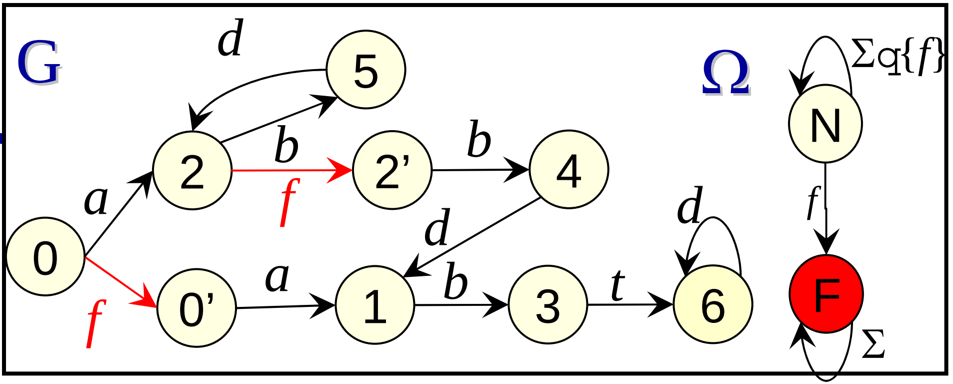
= Yes / [[μ]] » L_{Q_P}(Ω)

(B) Bounded: **<*n*, ;*s* 5 [[*μ*]] _ L_{Q_P}(Ω), ;*t* 5 *L*(*G*)/*s* _ Σ*.Σ**

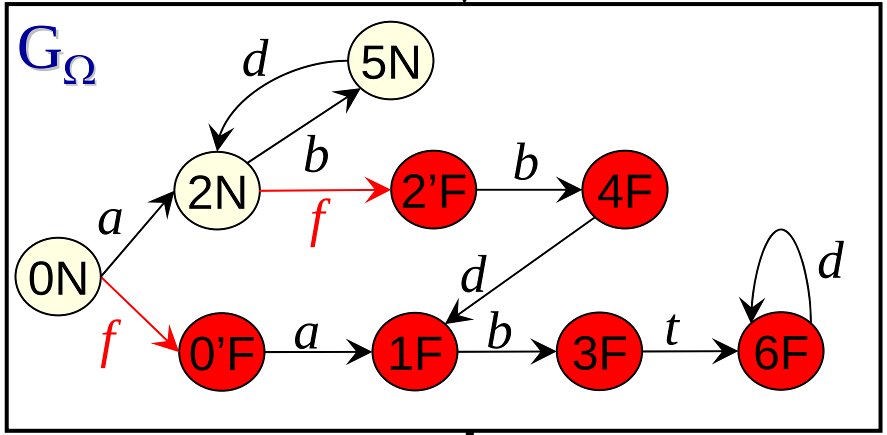
mmP(*t*)mm¿ *n* , Diag(μ.P(*t*))=Yes



Construction of the diagnoser



- ❑ We take Ω into account in *G*
  - ➢ G_Ω = GnΩ
    - ✓ $L(G_\Omega)=L(G)$
    - ✓ $L(G_\Omega,Q\S\ Q_P) = L(G)\ \_\ L_{Q_P}(\Omega)$



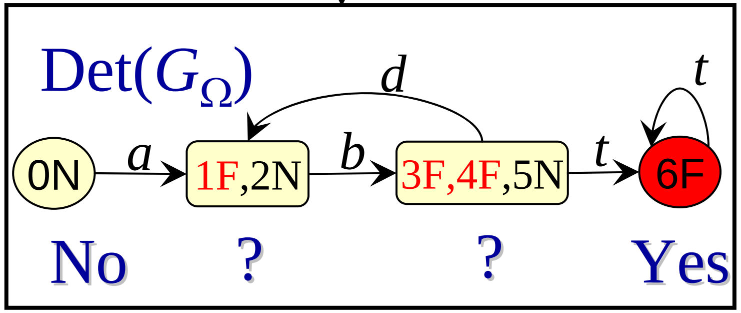
- ❑ Det(G_Ω) = kX, Σ_o ,X_o,$ $_d$ l,
  - ➢ X_o= (q0,q0$_\Omega$), X 5 $2^{Q\S\ Q_\Omega}$
  - ➢ $\Delta_{Det(G\Omega)}(X_0,\mu) = \{\Delta_{G_\Omega}((q0,q0_\Omega),[[\mu]])\}$

$\Delta_{DetG\Omega}(X_0,abd) = \{1F,2N\}$

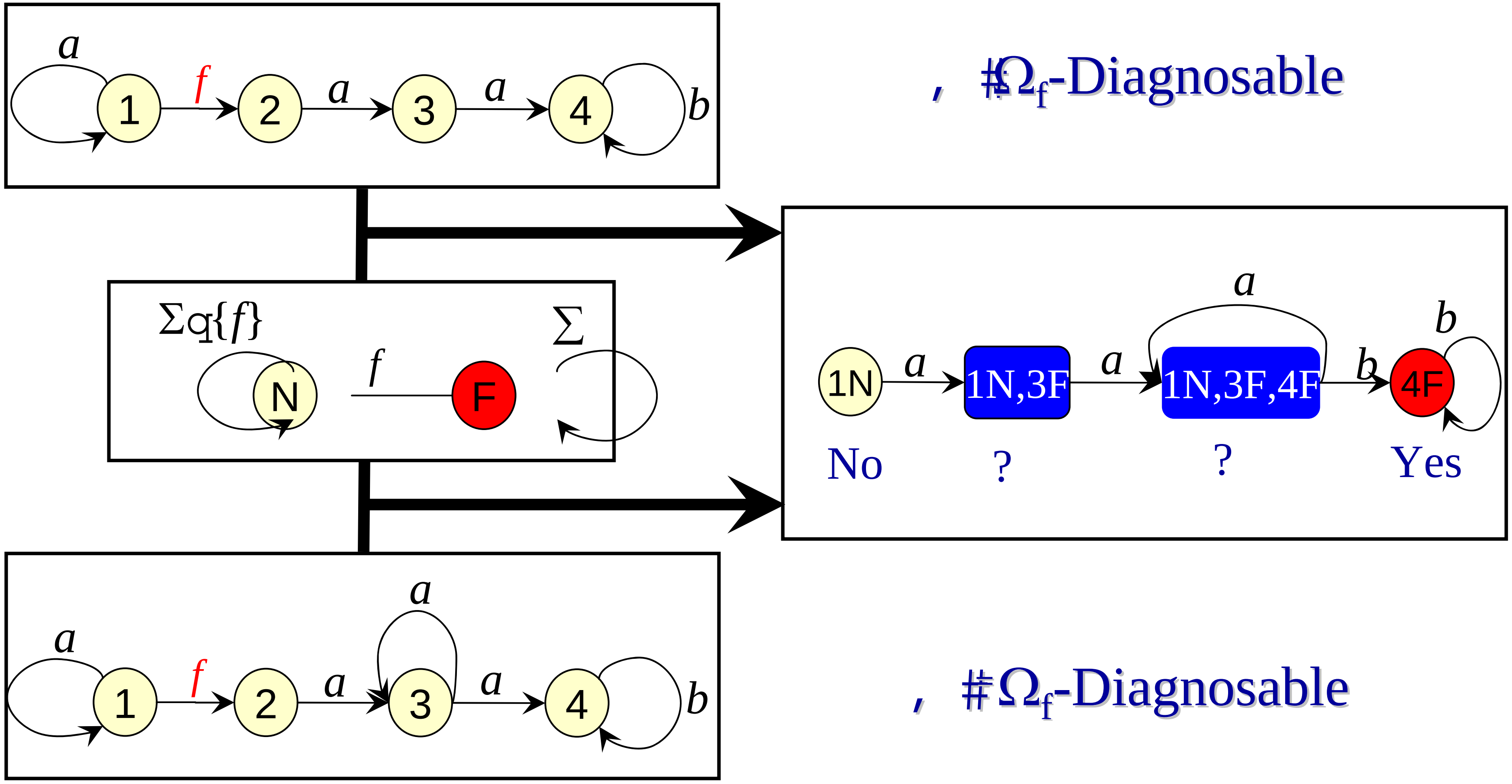
- ❑ The function Diag$_\Omega(\mu)$

$$Diag_\Omega(\mu)=\begin{cases} Yes & \text{si} \quad \Delta_{Det(G_\Omega)}(X_o,\mu)\subseteq Q\times Q_P \\ No & \text{si} \quad \Delta_{Det(G_\Omega)}(X_o,\mu)\cap Q\times Q_P=\emptyset \\ ? & \text{otherwise} \end{cases}$$

is a correct diagnoser



Diagnoser et Diagnosability



, #Ω$_f$-Diagnosable

, #Ω$_f$-Diagnosable

Test of the diagnosability

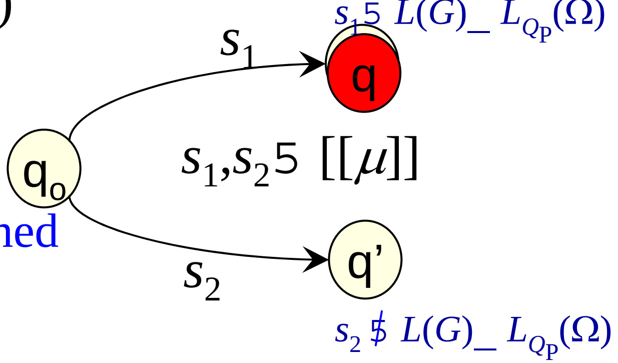
- ❑ *G* is not Ω-diagnosable if it exists 2 arbitrary long sequences compatible with the observation, one being faulty the other not. [Jiang 00 & Yoo 02]

- ❑ Test based on Obs(G_Ω), the $(\Sigma_{uo})^*\Sigma_o$-closure of G_Ω

- ❑ Γ = Obs(G_Ω) § Obs(G_Ω)

✓ $(q_0,q_0) \xrightarrow{\mu} (q,q')$,

Ω–undetermined



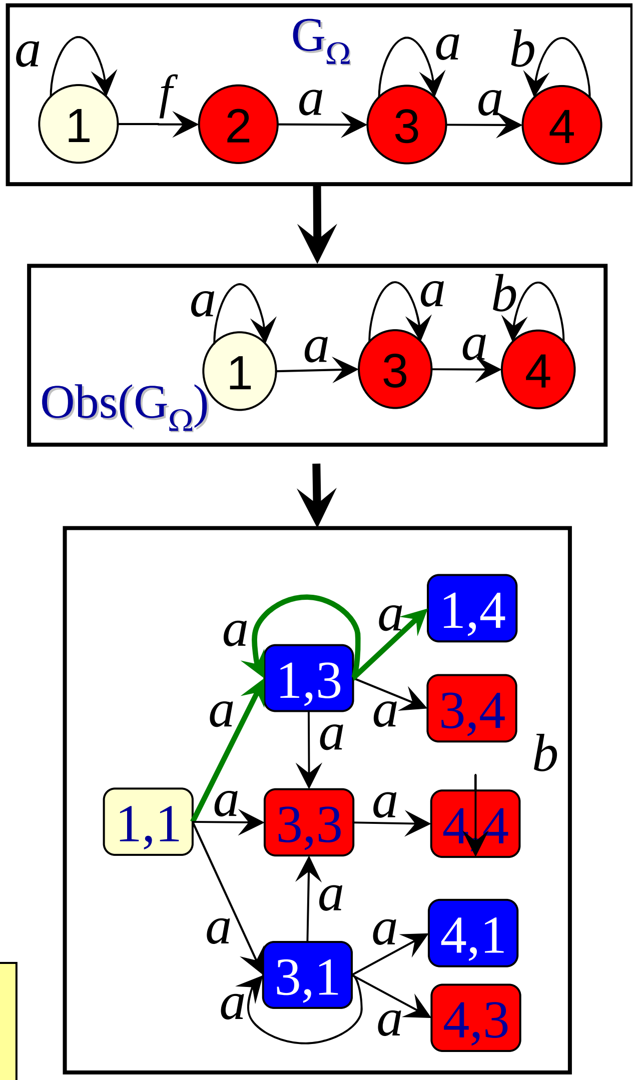
s_1 5 $L(G)$_ $L_{Q_P}(\Omega)$
s_1
q
q_o
$s_1,s_2$5 [[μ]]
s_2
q'
s_2 ∉ $L(G)$_ $L_{Q_P}(\Omega)$

✓ A path in Γ is *n*-undetermined

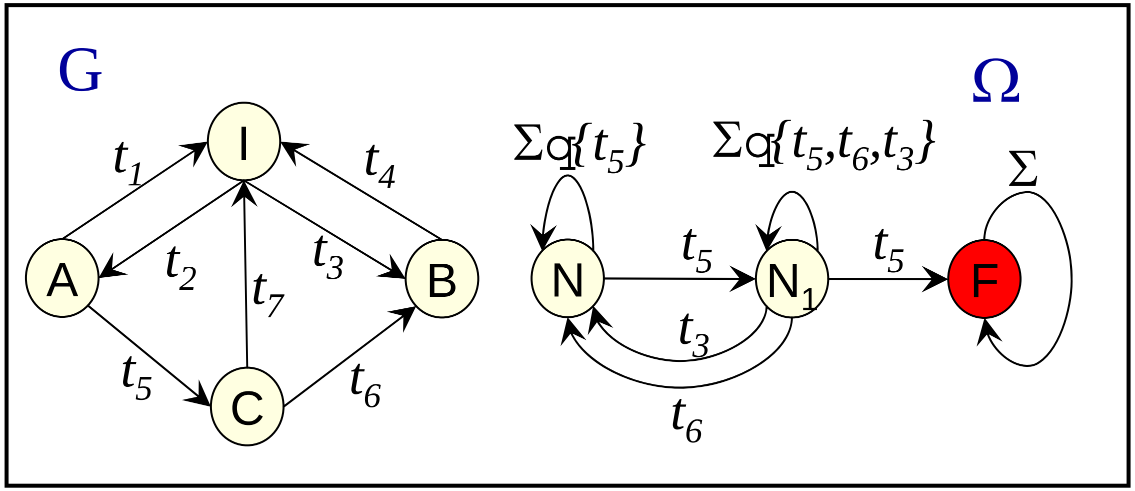
if it contains *n*+1 states Ω-undetermined

G is Ω(n)-diagnosable / ∄ path *n*-undetermined in Γ

G is Ω-diagnosable / no Ω-undetermined cycle in Γ



Example: supervision of a building



A : office (accueil)

I : laboratory

C : cafeteria

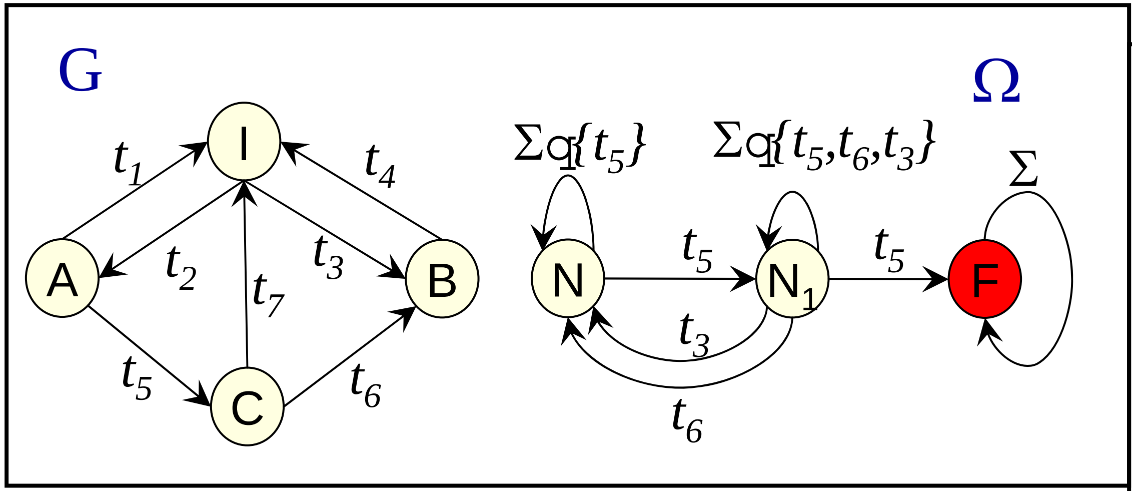
B : library

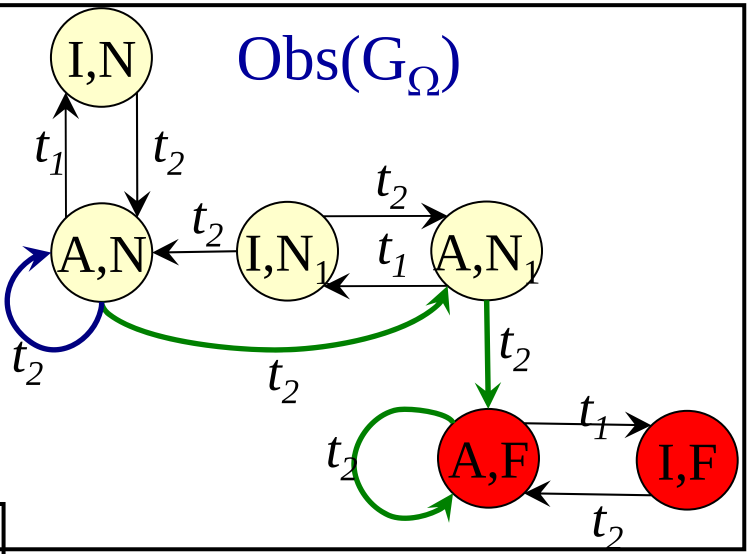
t1,t2, t3, t4, t5,t6,t7 : doors with magnetic cards (observable or not)

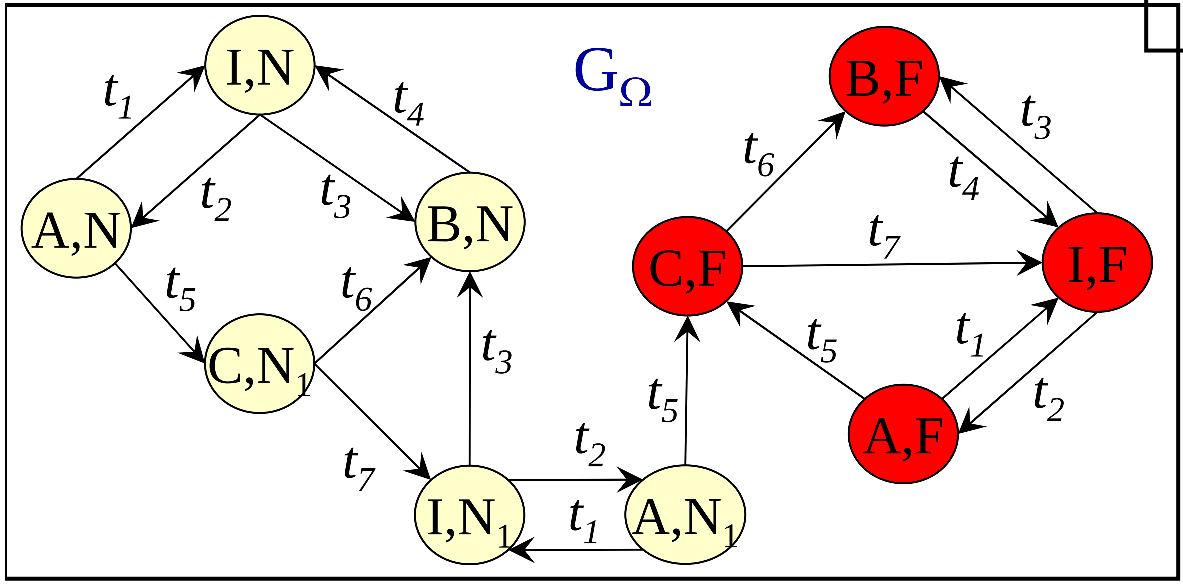
Property to checked :

“Having gone twice to the cafeteria without going through the library”

Example: supervision of a building



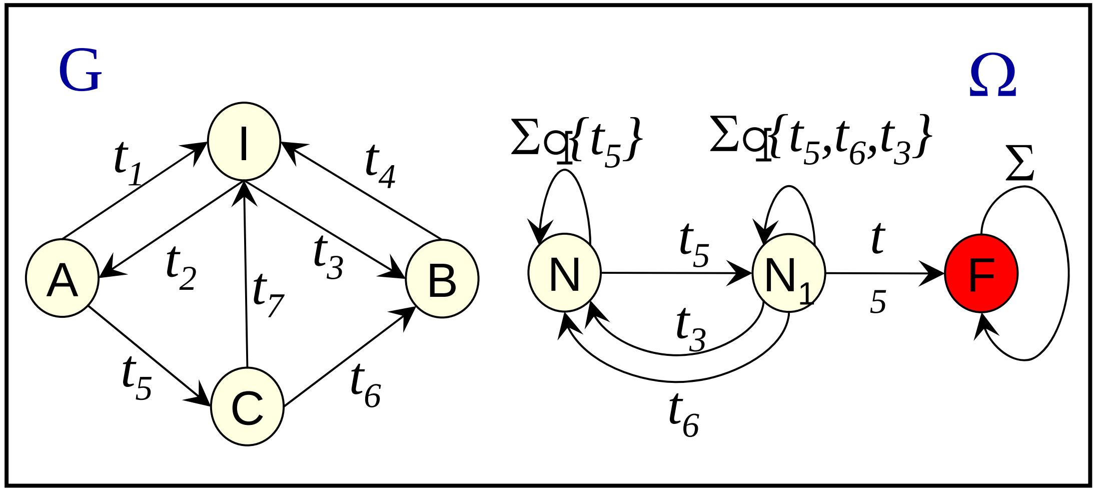


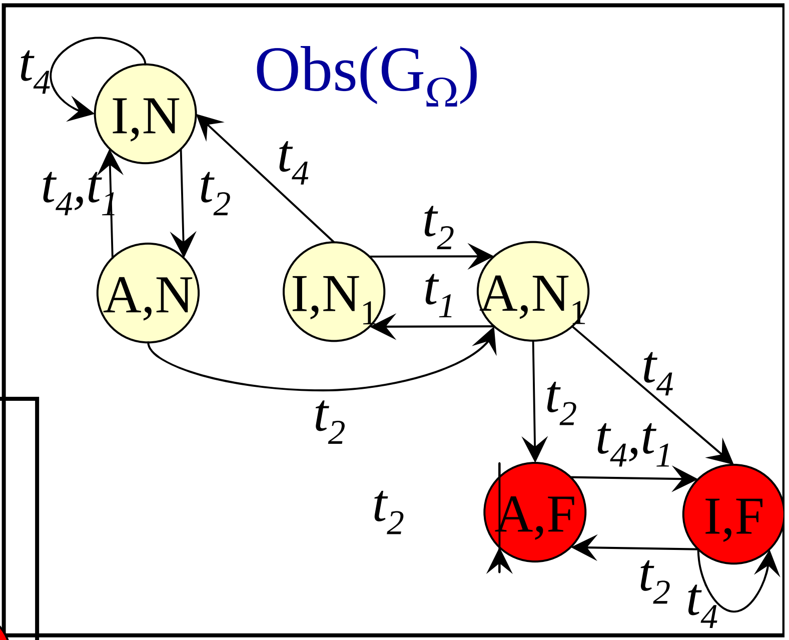


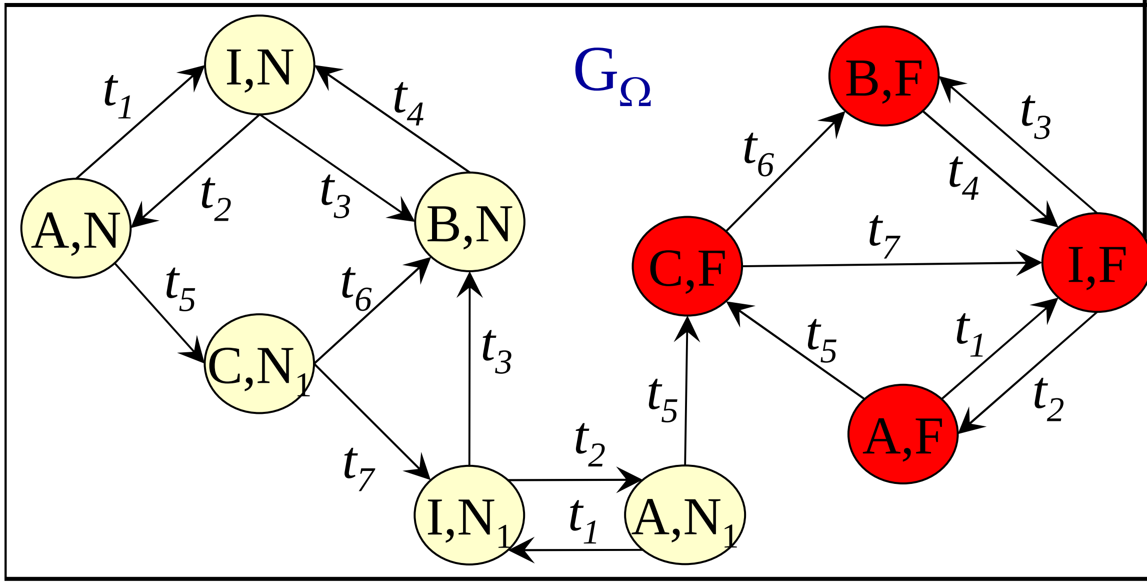
Not diagnosticable !

$\Sigma_o=\{t_1,t_2\}$

Example: supervision of a building







Diagnosticable !

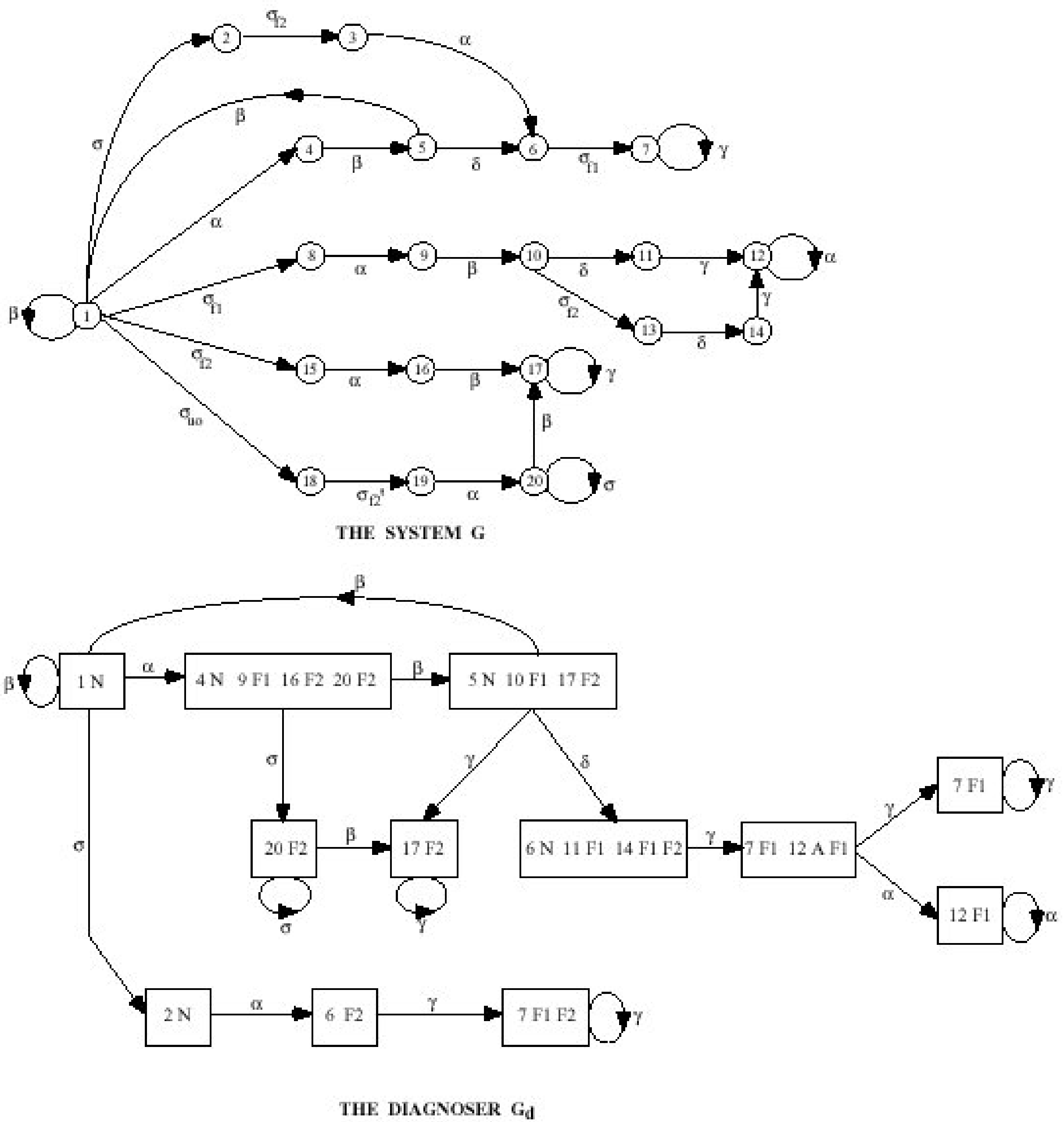
$\Sigma_o=\{t_1,t_2,t_4\}$

$\mathrm{Obs}(G_\Omega)=\mathrm{Det}(G_\Omega)$

Conclusion and perspectives on diagnosability

- Analysis of real large-scale discrete event systems
  - See Decentralized/distributed approaches of diagnosability
- Help to improve the efficiency of diagnosis
  - By choosing the good abstraction level for the model
- Help to improve the diagnosability of a system
  - By improving its observability : how to place the sensors for a better discrimination? Optimality of sensor cost versus failure cost?
- Help to design a self-healable system
  - Which repair actions for a system given its diagnosability?

Di



THE SYSTEM G

THE DIAGNOSER G_d

Figure 3.5: Example illustrating construction of the diagnoser G_d

IAF'08 – GDR I3 – 21 octobre 2008, Paris Dauphine

Systèmes autoguérissants : vers une intégration formelle du diagnostic et de la réparation

Marie-Odile Cordier, Thierry Vidal DREAM IRISA

Yannick Pencolé, Louise Travé-Massuyès DISCO LAAS

Hypothèses de travail et objectifs

- Système continu ou **discret**
- **Centralisé**
- **Pas de contraintes de temps**
- Fautes **permanentes** et **uniques**
- Notion **abstraite** de plans de réparation

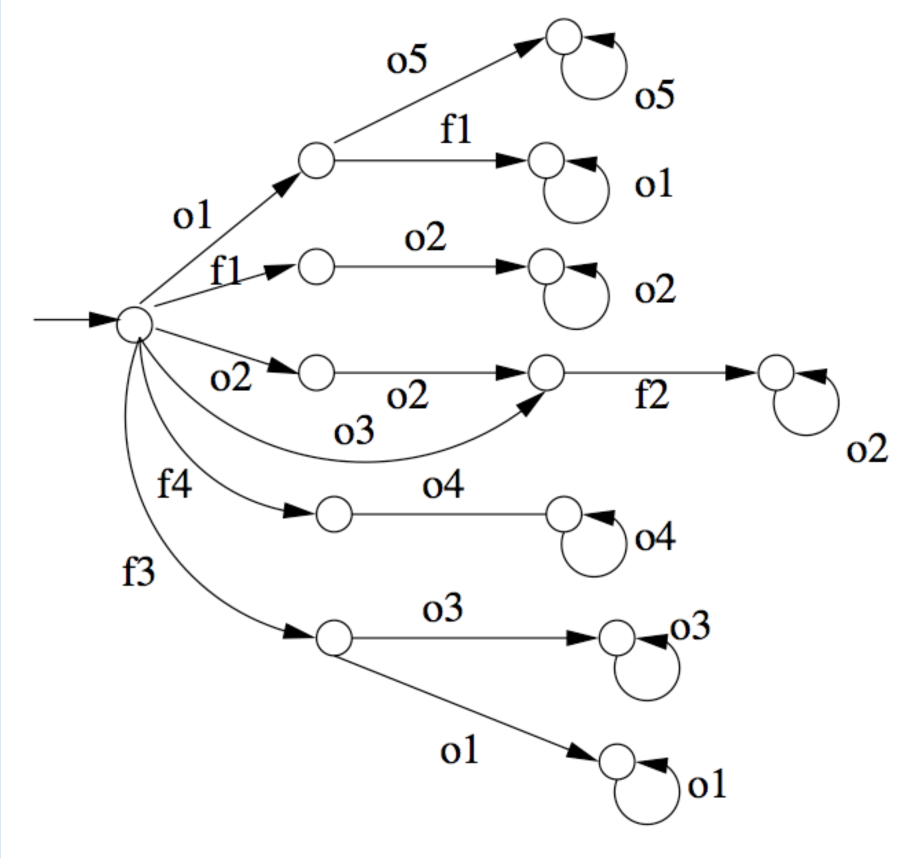
… amenées à être relaxées !

- Chercher des **propriétés formelles génériques**
- Définitions « **intégrées** » : diagnosticabilité + réparabilité = autoguérison

Diagnosticabilité

o_1 ,..., o_5 observables

f_1 ,..., f_4 fautes non observables



Diagnosticabilité

o_1 ,…, o_5 observables

f_1 ,…, f_4 fautes non observables

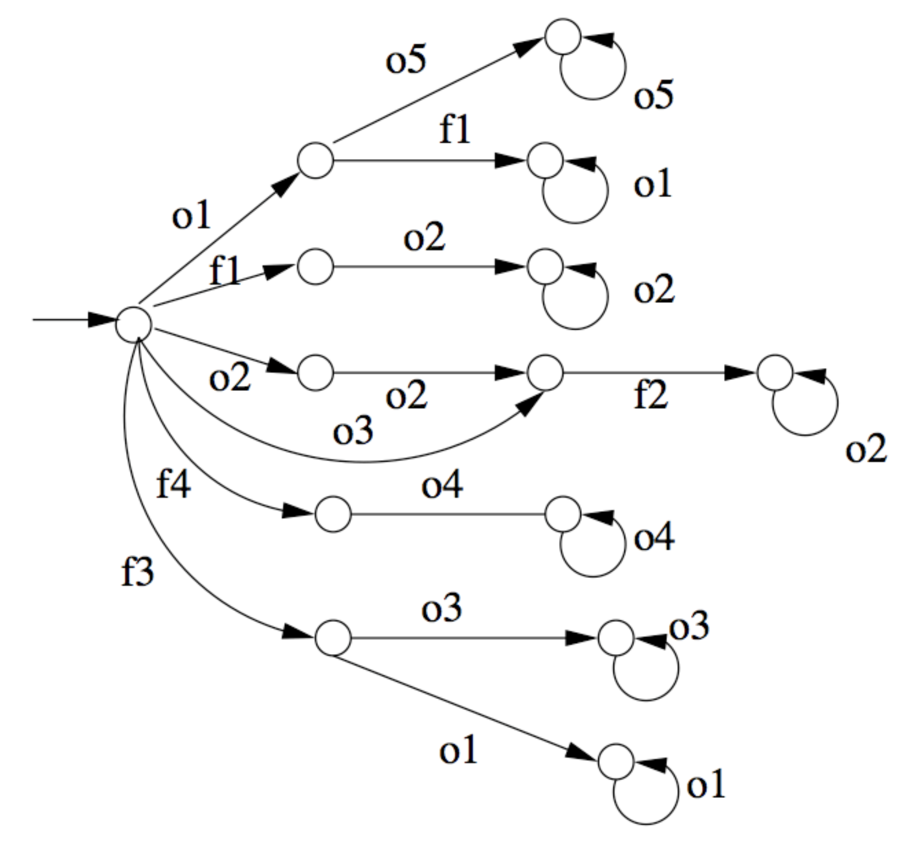
Sig(f_1): $\{o_1^\infty, o_2^\infty\}$

Sig(f_2): $\{o_2^\infty, o_3o_2^\infty\}$

Sig(f_3): $\{o_1^\infty, o_3^\infty\}$

Sig(f_4): $\{o_4^\infty\}$

Sig(ok): $\{o_1o_5^\infty\}$



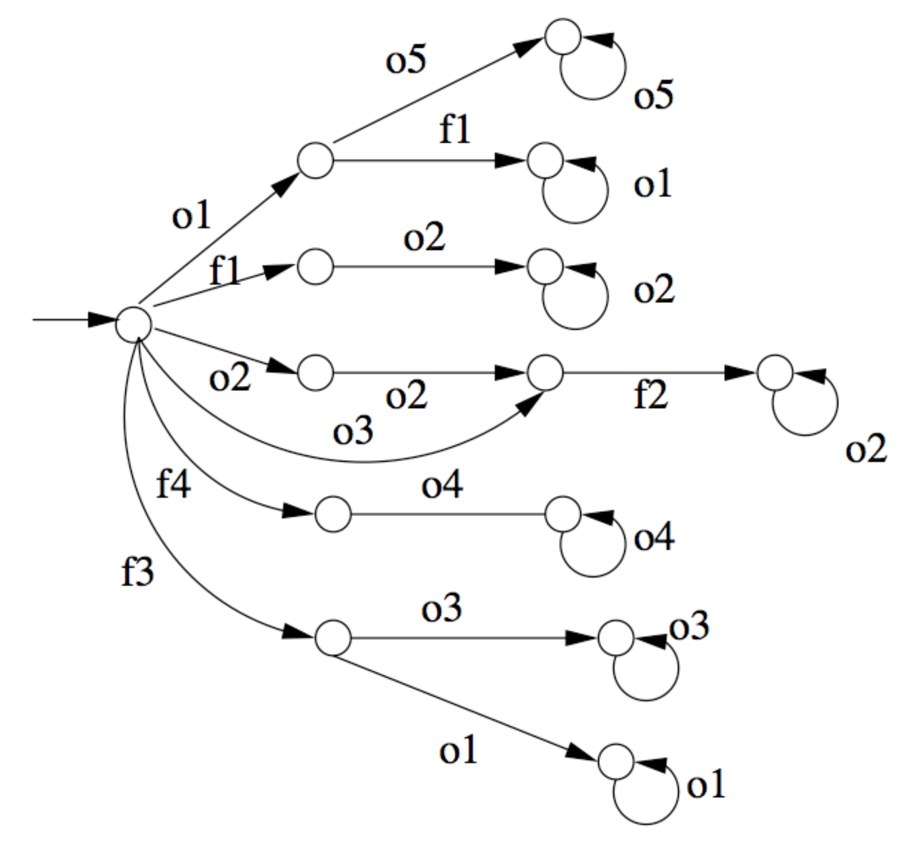
Diagnosticabilité

o_1 ,…, o_5 observables

f_1 ,…, f_4 fautes non observables

Sig(f_1): {o_1^∞, o_2^∞}
Sig(f_2): {o_2^∞, $o_3o_2^\infty$}
Sig(f_3): {o_1^∞, o_3^∞}
Sig(f_4): {o_4^∞}
Sig(ok): {$o_1o_5^\infty$}

{{ok},{f_1,f_2,f_3},{f_4}}
{{ok},{f_1,f_2},{f_1,f_3},{f_4}}
{{ok},{f_1,f_2},{f_1,f_3},{f_2},{f_3},{f_4}}
diagnosticables !



Diagnosticabilité

o_1 ,…, o_5 observables

f_1 ,…, f_4 fautes non observables

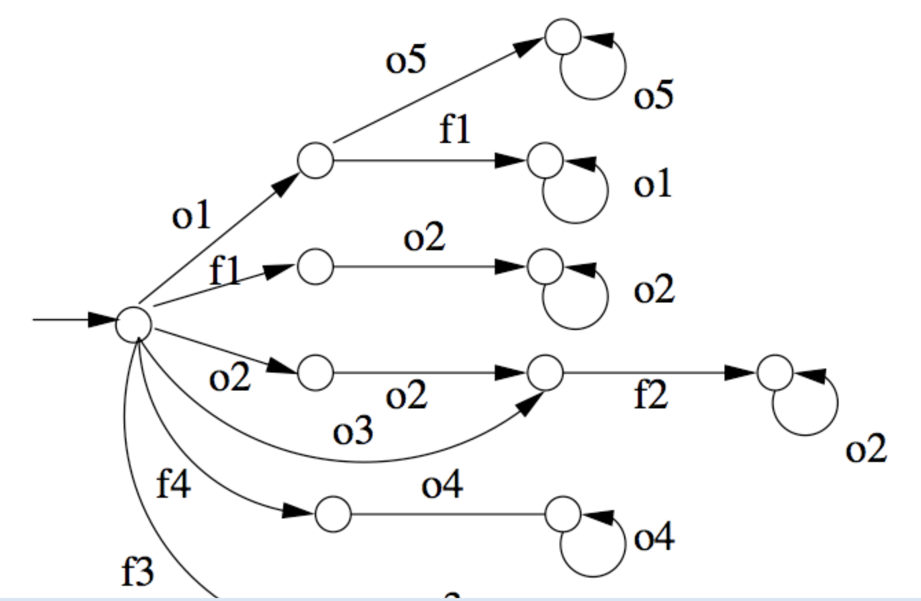
Sig(f_1): $\{o_1^\infty, o_2^\infty\}$

Sig(f_2): $\{o_2^\infty, o_3o_2^\infty\}$

Sig(f_3): $\{o_1^\infty, o_3^\infty\}$

Sig(f_4): $\{o_4^\infty\}$

Sig(ok): $\{o_1o_5^\infty\}$



$\{\{ok\},\{f_1,f_2,f_3\},\{f_4\}\}$

$\{\{ok\},\{f_1,f_2\},\{f_1,f_3\},\{f_4\}\}$

$\{\{ok\},\{f_1,f_2\},\{f_1,f_3\},\{f_2\},\{f_3\},\{f_4\}$

diagnosticables !

$\{\{o_1o_5^\infty\},\{o_1^\infty,o_2^\infty,o_3o_2^\infty,o_3^\infty\},\{o_4^\infty\}\}$

$\{\{o_1o_5^\infty\},\{o_2^\infty,o_3o_2^\infty\},\{o_1^\infty, o_3^\infty\},\{o_4^\infty\}\}$

$\{\{o_1o_5^\infty\},\{o_2^\infty\},\{o_1^\infty\},\{o_3o_2^\infty\},\{o_3^\infty\},$
$\{o_4^\infty\}\}$

Diagnosticabilité

- $\mathcal{F}$: ensemble de fautes élémentaires (+ ok)
- $E(\mathcal{F}) = \{F_1, F_2, \dots, F_m\}$: ensemble de macro-fautes
- Ensemble d'observables Σ -> $MF(\Sigma)$ macro-faute minimale produisant Σ

Diagnosticable (E($\mathcal{F}$)) <=>

- ∃ *partition des observables* $\pi = \{\Sigma_1, \dots, \Sigma_m\}$
- $\forall i$, $MF(\Sigma_i) = F_i$

Réparabilité

$\{\{ok\},\{f_1,f_2,f_3\},\{f_4\}\}$ réparable

ssi

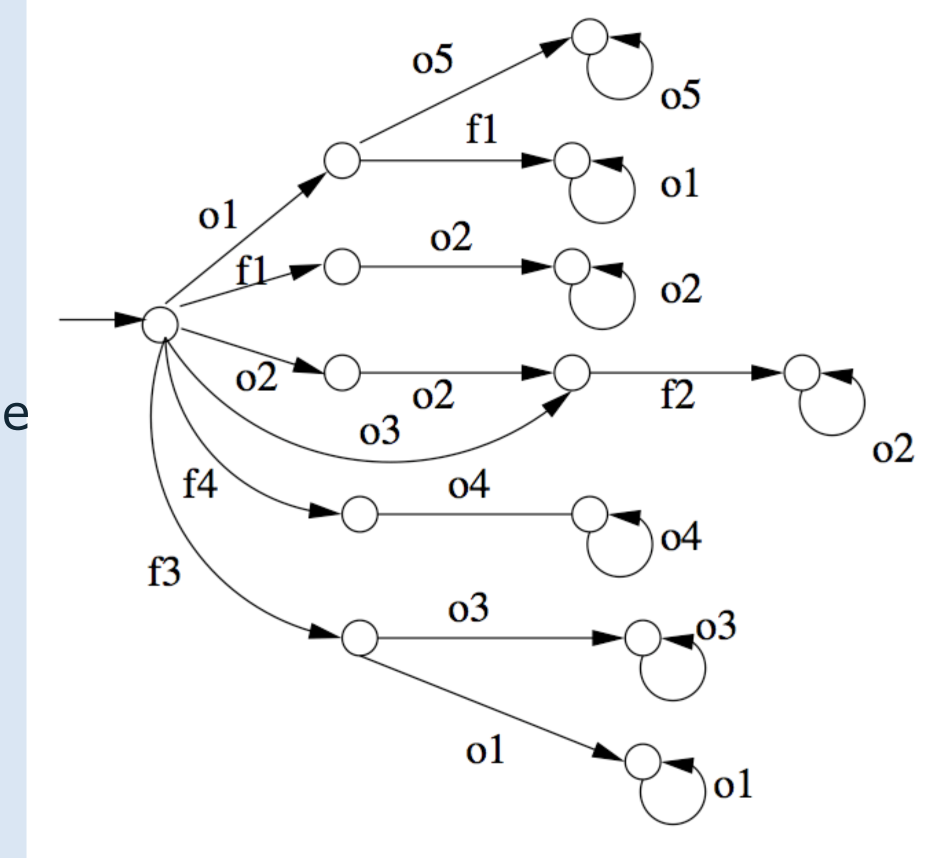
$\exists r_a$ réparant f_1, f_2, et f_3

$\{\{ok\},\{f_1,f_2\},\{f_1,f_3\},\{f_4\}\}$ réparable

ssi

$\exists r_a$ réparant $f_1$et f_2, et

$\exists r_b$ réparant $f_1$et f_3



Réparabilité & Autoguérison

Réparable (E($\mathcal{F}$)) <=>∀ F_i , ∃ r_k / Répare(r_k , F_i)

Autoguérissant <=>

- ∃ *E($\mathcal{F}$) couvrant*
- *Diagnosticable(E($\mathcal{F}$))*
- *Réparable(E($\mathcal{F}$))*

Algo sur l’exemple

	O_1O_5 ∞	$O_1\infty$	$O_2\infty$	O_3O_2 ∞	$O_3\infty$	$O_4\infty$
ok						
f_1						
f_2						
f_3						
f_4						

Algo sur l’exemple

	O_1O_5 ∞	O_1^∞	O_2^∞	O_3O_2 ∞	O_3^∞	O_4^∞
ok	X					
f_1		X	X			
f_2			X	X		
f_3		X			X	
f_4						X

- Relier les fautes élémentaires à leurs signatures

Algo sur l'exemple

	O_1O_5 ∞	O_1^∞	O_2^∞	O_3O_2 ∞	O_3^∞	O_4^∞
ok	r_{ok}					
f_1		r_1 r_2	r_1 r_2			
f_2			r_1 r_3	r_1 r_3		
f_3		r_2 r_3			r_2 r_3	
f_4						r_4

- Expliciter les plans qui réparent la faute

Algo sur l'exemple

	$O_1O_5^{\infty}$	O_1^{∞}	O_2^{∞}	$O_3O_2^{\infty}$	O_3^{∞}	O_4^{∞}
ok	r_{ok}					
f_1		r_1 r_2	r_1 r_2			
f_2			r_1 r_3	r_1 r_3		
f_3		r_2 r_3			r_2 r_3	
f_4						r_4

- Expliciter les plans qui réparent la faute
- … et établir l'intersection pour chaque signature

Algo sur l'exemple

	O_1O_5 ∞	O_1^∞	O_2^∞	O_3O_2 ∞	O_3^∞	O_4^∞
ok	r_{ok}					
f_1		~~r_1~~ r_2	r_1 ~~r_2~~			
f_2			r_1 ~~r_3~~	r_1 r_3		
f_3		r_2 ~~r_3~~			r_2 r_3	
f_4						r_4

- Expliciter les plans qui réparent la faute
- … et établir l'intersection pour chaque signature

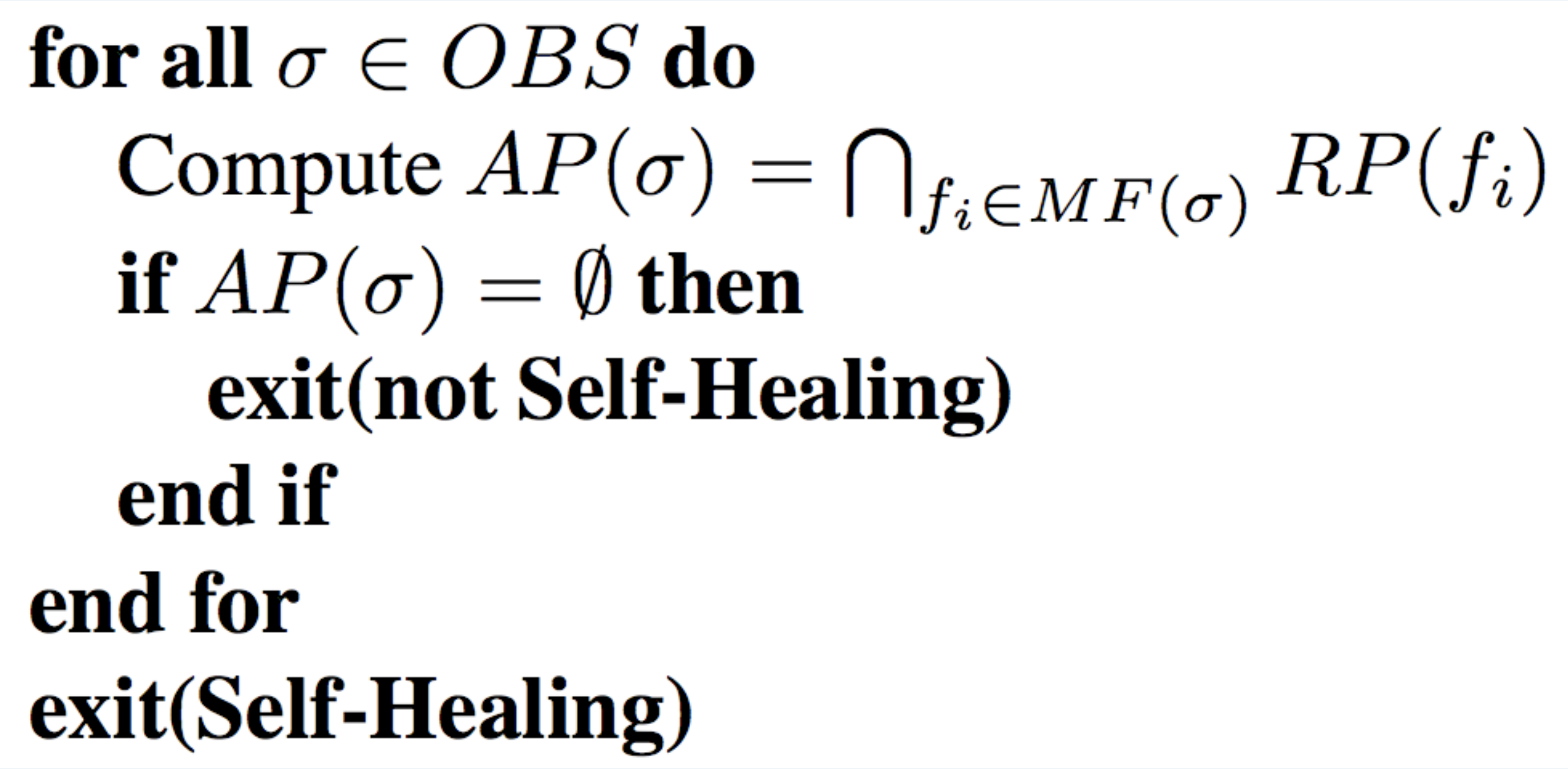
Algo sur l’exemple

	O_1O_5 ∞	O_1^∞	O_2^∞	O_3O_2 ∞	O_3^∞	O_4^∞
ok	r_{ok}					
f_1		~~r_1~~ r_2	r_1 ~~r_2~~			
f_2			r_1 ~~r_3~~	r_1 r_3		
f_3		r_2 ~~r_3~~			r_2 r_3	
f_4						r_4
	r_{ok}	**r_2**	**r_1**	**r_1 r_3**	**r_2 r_3**	**r_4**

Algo sur l'exemple

	O_1O_5 ∞	O_1∞	O_2∞	O_3O_2 ∞	O_3∞	O_4∞
ok	r_{ok}					
f_1		~~r_1~~ r_2	r_1 ~~r_2~~			
f_2			r_1 ~~r_3~~	r_1 r_3		
f_3		r_2 ~~r_3~~			r_2 r_3	
f_4						r_4
E(B)=	**r_{ok}** { {ok}	**r_2** $\{f_1,f_3\}$	**r_1** $\{f_1,f_2\}$	**r_1 r_3** $\{f_2\}$	**r_2 r_3** $\{f_3\}$	**r_4** $\{f_4\}$ }

Algo de vérification de SH



```
for all $\sigma \in OBS$ do
    Compute $AP(\sigma) = \bigcap_{f_i \in MF(\sigma)} RP(f_i)$
    if $AP(\sigma) = \emptyset$ then
        exit(not Self-Healing)
    end if
end for
exit(Self-Healing)
```

Justification formelle

Considérons un E(F) spécifique :

$\mathbf{E_0}(\mathcal{F})=\bigcup_{\sigma\in OBS}\mathrm{MF}(\sigma)$

On prouve :

1. **Diagnosticable($\mathbf{E_0}(\mathcal{F})$)**
2. **Autoguérissant** ssi **Réparable($\mathbf{E_0}(\mathcal{F})$)**
3. L’algo vérifie justement **Réparable($\mathbf{E_0}(\mathcal{F})$)**

Sélection des plans

O_1O_5 ∞	O_1 ∞	O_2 ∞	O_3O_2 ∞	O_3 ∞	O_4 ∞
$\mathbf{r_{ok}}$	$\mathbf{r_2}$	$\mathbf{r_1}$	$\mathbf{r_1}$, $\mathbf{r_3}$	$\mathbf{r_2}$, $\mathbf{r_3}$	$\mathbf{r_4}$

E(B)= $\{ \{ok\}$ $\{f_1,f_3\}$ $\{f_1,f_2\}$ $\{f_2\}$ $\{f_3\}$ $\{f_4\} \}$

Sélection des plans

E(B)=

$O_1O_5^\infty$	O_1^∞	O_2^∞	$O_3O_2^\infty$	O_3^∞	O_4^∞
r_{ok}	r_2	r_1	r_1, r_3	r_2, r_3	r_4
{ {ok}	$\{f_1,f_3\}$	$\{f_1,f_2\}$	$\{f_2\}$	$\{f_3\}$	$\{f_4\}$ }

$O_1O_5^\infty$	O_1^∞, O_3^∞	$O_2^\infty, O_3O_2^\infty$	O_4^∞
r_{ok}	r_2	r_1	r_4

Sélection des plans

$O_1O_5^\infty$	O_1^∞	O_2^∞	$O_3O_2^\infty$	O_3^∞	O_4^∞
r_{ok}	r_2	r_1	r_1, r_3	r_2, r_3	r_4

E(B)= { {ok} {f_1,f_3} {f_1,f_2} {f_2} {f_3} {f_4} }

$O_1O_5^\infty$	O_1^∞ ,O_3^∞	O_2^∞ ,$O_3O_2^\infty$	O_4^∞
r_{ok}	r_2	r_1	r_4

E(B)= { {ok} {f_1,f_3} {f_1,f_2} {f_4} }

Sélection des plans

	O_1O_5 ∞	O_1∞	O_2∞	O_3O_2 ∞	O_3∞	O_4∞
E(B)=	r_{ok} { {ok}	r_2 {f_1,f_3}	r_1 {f_1,f_2}	r_1 r_3 {f_2}	r_2 r_3 {f_3}	r_4 {f_4} }

Sélection des plans

E(B)=

O_1O_5 ∞	$O_1{}^{\infty}$	$O_2{}^{\infty}$	O_3O_2 ∞	$O_3{}^{\infty}$	$O_4{}^{\infty}$
r_{ok} { {ok}}	r_2 $\{f_1,f_3\}$	r_1 $\{f_1,f_2\}$	$\cancel{r_1}, r_3$ $\{f_2\}$	$\cancel{r_2}, r_3$ $\{f_3\}$	r_4 $\{f_4\}$ }

O_1O_5 ∞	$O_1{}^{\infty}$	$O_2{}^{\infty}$	$O_3O_2{}^{\infty},O_3{}^{\infty}$	$O_4{}^{\infty}$
r_{ok}	r_2	r_1	r_3	r_4

Sélection des plans

	O_1O_5 ∞	O_1^∞	O_2^∞	O_3O_2 ∞	O_3^∞	O_4^∞
E(B)=	$\mathbf{r_{ok}}$ { {ok} }	$\mathbf{r_2}$ $\{f_1,f_3\}$	$\mathbf{r_1}$ $\{f_1,f_2\}$	$\cancel{\mathbf{r_1}}$, $\mathbf{r_3}$ $\{f_2\}$	$\cancel{\mathbf{r_2}}$, $\mathbf{r_3}$ $\{f_3\}$	$\mathbf{r_4}$ $\{f_4\}$ }

	O_1O_5 ∞	O_1^∞	O_2^∞	$O_3O_2^\infty, O_3^\infty$	O_4^∞
E(B)=	$\mathbf{r_{ok}}$ { {ok} }	$\mathbf{r_2}$ $\{f_1,f_3\}$	$\mathbf{r_1}$ $\{f_1,f_2\}$	$\mathbf{r_3}$ $\{f_2 ,f_3\}$	$\mathbf{r_4}$ $\{f_4\}$ }

Si le système n'est pas SH ?

	$O_1O_5^{\infty}$	O_1^{∞}	O_2^{∞}	$O_3O_2^{\infty}$	O_3^{∞}	O_4^{∞}
ok	r_{ok}					
f_1		r_1	r_1			
f_2			r_1 r_3	r_1 r_3		
f_3		r_2 r_3			r_2 r_3	
f_4						r_4
	$\mathbf{r_{ok}}$	🟥	$\mathbf{r_1}$	$\mathbf{r_1}$ $\mathbf{r_3}$	$\mathbf{r_2}$ $\mathbf{r_3}$	$\mathbf{r_4}$

SH faible
= pour un sous-ensemble des σ_i

	$O_1O_5^\infty$	[redacted]	O_2^∞	$O_3O_2^\infty$	O_3^∞	O_4^∞
ok	r_{ok}	[redacted]				
f_1		[redacted]	r_1			
f_2		[redacted]	r_1 r_3	r_1 r_3		
f_3		[redacted]			r_2 r_3	
f_4		[redacted]				r_4
	$\mathbf{r_{ok}}$	[redacted]	$\mathbf{r_1}$	$\mathbf{r_1}$ $\mathbf{r_3}$	$\mathbf{r_2}$ $\mathbf{r_3}$	$\mathbf{r_4}$

SH faible

= pour un sous-ensemble des σ_i

	$O_1O_5^\infty$	[redacted]	O_2^∞	$O_3O_2^\infty$	O_3^∞	O_4^∞
ok	r_{ok}	[redacted]				
f_1		[redacted]	r_1			
f_2		[redacted]	r_1 $\cancel{r_3}$	r_1 r_3		
f_3		[redacted]			r_2 r_3	
f_4		[redacted]				r_4
E(B)=	$\mathbf{r_{ok}}$ { {ok}	[redacted]	$\mathbf{r_1}$ $\{f_1,f_2\}$	$\mathbf{r_1}$ $\mathbf{r_3}$ $\{f_2\}$	$\mathbf{r_2}$ $\mathbf{r_3}$ $\{f_3\}$	$\mathbf{r_4}$ $\{f_4\}$ }

SH partielle
= pour un sous-ensemble des f_i

	$O_1O_5^\infty$	O_1^∞	O_2^∞	$O_3O_2^\infty$	O_3^∞	O_4^∞
ok	r_{ok}					
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
f_2			r_1 r_3	r_1 r_3		
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
f_4						r_4
	$\mathbf{r_{ok}}$		$\mathbf{r_1}$	$\mathbf{r_1}$ $\mathbf{r_3}$	$\mathbf{r_2}$ $\mathbf{r_3}$	$\mathbf{r_4}$

SH partielle
= pour un sous-ensemble des f_i

	O_1O_5 ∞	O_1∞	O_2∞	O_3O_2 ∞	O_3∞	O_4∞
ok	r_{ok}					
f_1		r_1	r_1			
f_2			r_1 r_3	r_1 r_3		
f_3		r_2 r_3			r_2 r_3	
f_4						r_4
E(B)=	$\mathbf{r_{ok}}$ { {ok}		$\mathbf{r_1}$ {f_2}	$\mathbf{r_1}$ $\mathbf{r_3}$ {f_2}	$\mathbf{r_2}$ $\mathbf{r_3}$	$\mathbf{r_4}$ {f_4} }

Rétablir la SH ?

	O_1O_5 ∞	O_1^∞	O_2^∞	O_3O_2 ∞	O_3^∞	O_4^∞
ok	r_{ok}					
f_1		~~r_1~~	~~r_1~~			
f_2			r_1 r_3	r_1 r_3		
f_3		~~r_2~~ ~~r_3~~			r_2 r_3	
f_4						r_4
	$\mathbf{r_{ok}}$	🟥	$\mathbf{r_1}$	$\mathbf{r_1}$ $\mathbf{r_3}$	$\mathbf{r_2}$ $\mathbf{r_3}$	$\mathbf{r_4}$

Rétablir la SH ?

	O_1O_5 ∞	O_1^∞	O_2^∞	O_3O_2	O_3^∞	O_4^∞
ok	r_{ok}					
f_1	~~r_1~~		~~r_1~~			
f_2			r_1 r_3	r_1 r_3		
f_3		~~r_2~~ ~~r_3~~			r_2 r_3	
f_4						r_4
	$\mathbf{r_{ok}}$	🟥	$\mathbf{r_1}$	$\mathbf{r_1}$ $\mathbf{r_3}$	$\mathbf{r_2}$ $\mathbf{r_3}$	$\mathbf{r_4}$

1) Nouveaux plans de réparation

ajouter r_4 qui répare f_1 et f_3

Rétablir la SH ?

	$O_1O_5^\infty$	O_1^∞	O_2^∞	O_3O_2	O_3^∞	O_4^∞
ok	r_{ok}					
f_1		r_1				
f_2						
f_3		r_2 r				
f_4						r_4
	$\mathbf{r_{ok}}$	🟥	$\mathbf{r_1}$	$\mathbf{r_1\ r_3}$	$\mathbf{r_2\ r_3}$	$\mathbf{r_4}$

1) Nouveaux plans de réparation

ajouter r_4 qui répare f_1 et f_3

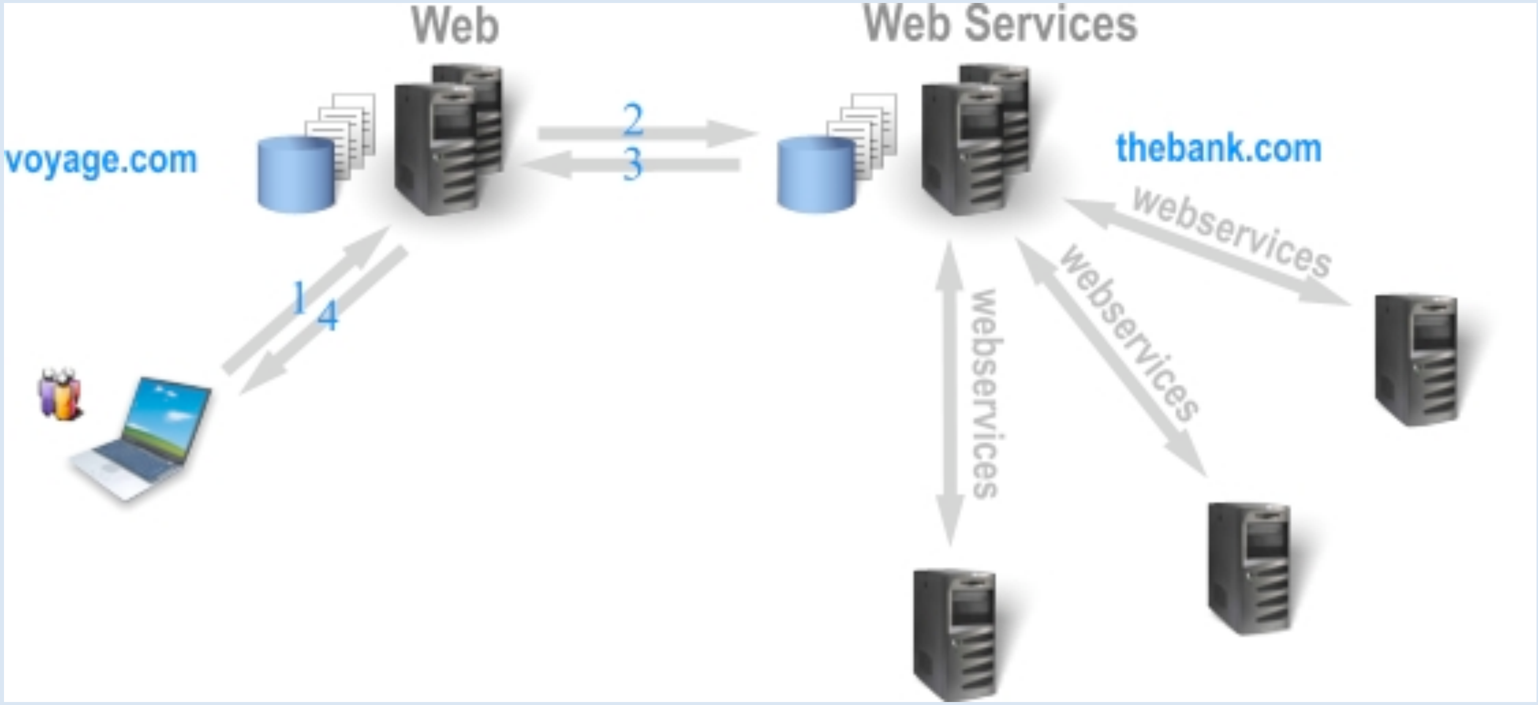
2) Nouveaux capteurs

ajouter des observations précisant les deux trajectoires donnant O_1^∞

Une architecture de diagnostic et réparation distribués

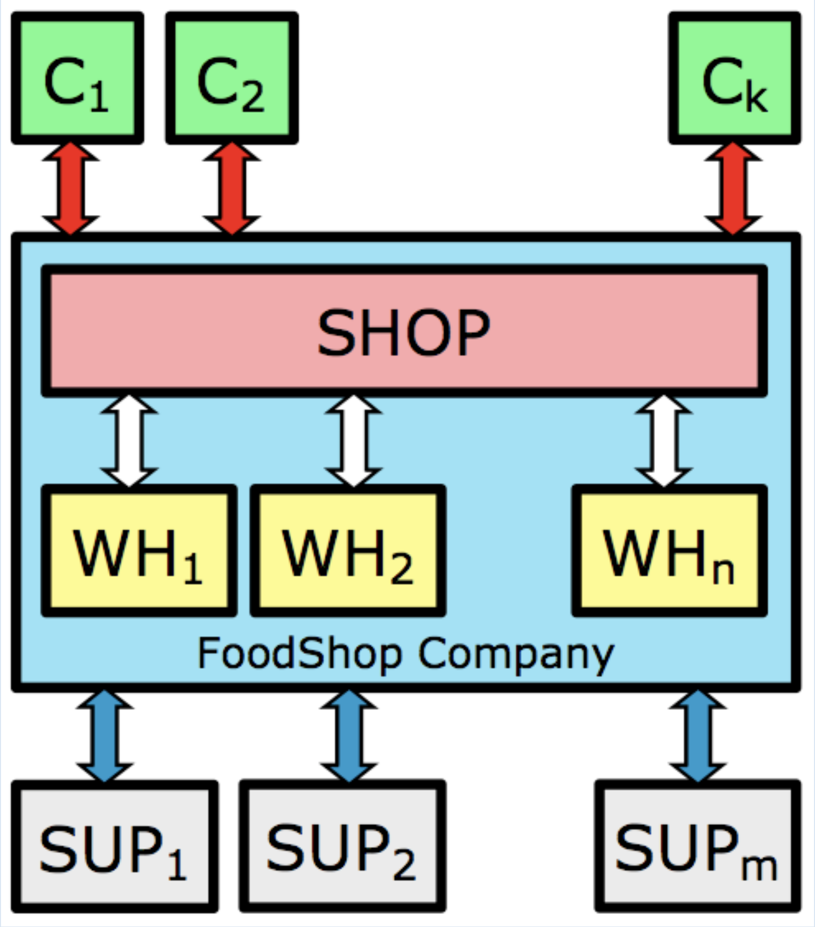
Le cadre applicatif

- WS-DIAMOND = Web-Service Diagnosis, Monitoring and Diagnosability



Le cadre applicatif

- Exemple : Food Shop

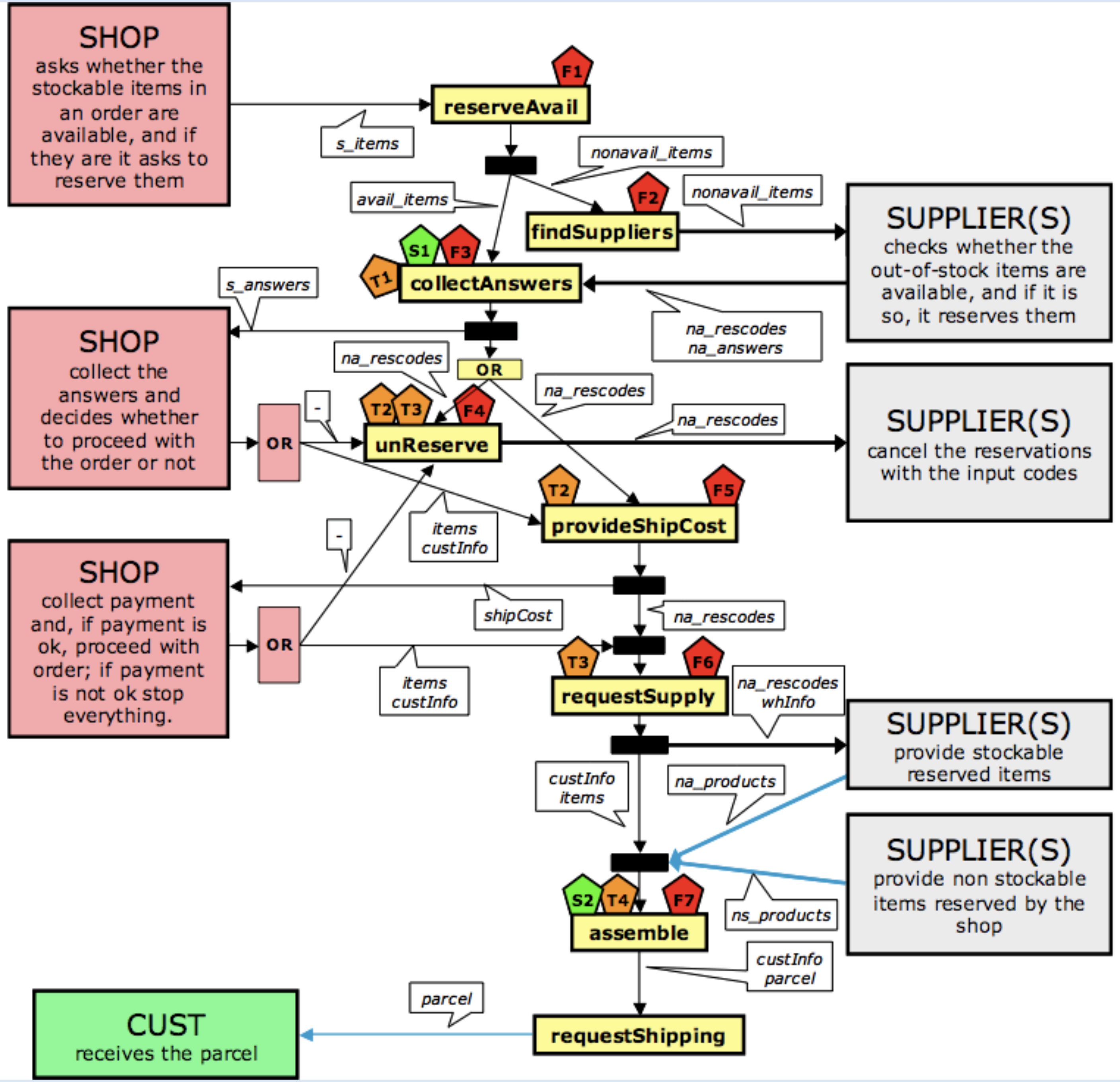


Le cadre applicatif

- Requête → exécution de process = workflow

Le c

- Req



SHOP
asks whether the stockable items in an order are available, and if they are it asks to reserve them

reserveAvail

F1

s_items

nonavail_items

avail_items

F2

nonavail_items

findSuppliers

SUPPLIER(S)
checks whether the out-of-stock items are available, and if it is so, it reserves them

S1

F3

T1

collectAnswers

s_answers

na_rescodes
na_answers

SHOP
collect the answers and decides whether to proceed with the order or not

na_rescodes

OR

na_rescodes

-

T2

T3

F4

na_rescodes

OR

unReserve

SUPPLIER(S)
cancel the reservations with the input codes

T2

F5

items
custInfo

provideShipCost

-

SHOP
collect payment and, if payment is ok, proceed with order; if payment is not ok stop everything.

shipCost

na_rescodes

OR

items
custInfo

T3

F6

requestSupply

na_rescodes
whInfo

SUPPLIER(S)
provide stockable reserved items

custInfo
items

na_products

SUPPLIER(S)
provide non stockable items reserved by the shop

S2

T4

F7

ns_products

assemble

custInfo
parcel

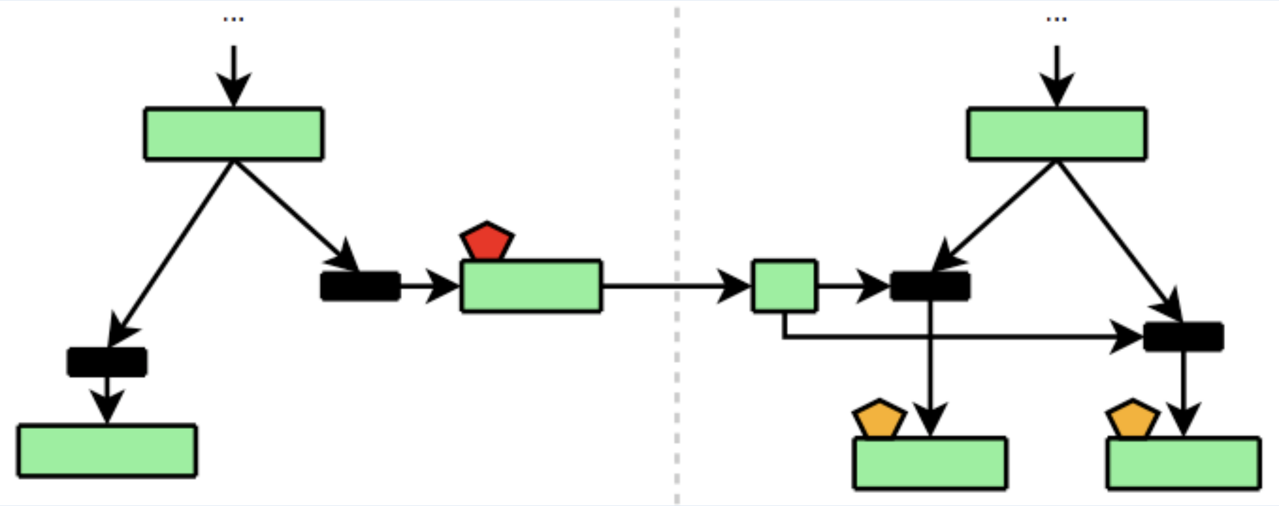
parcel

CUST
receives the parcel

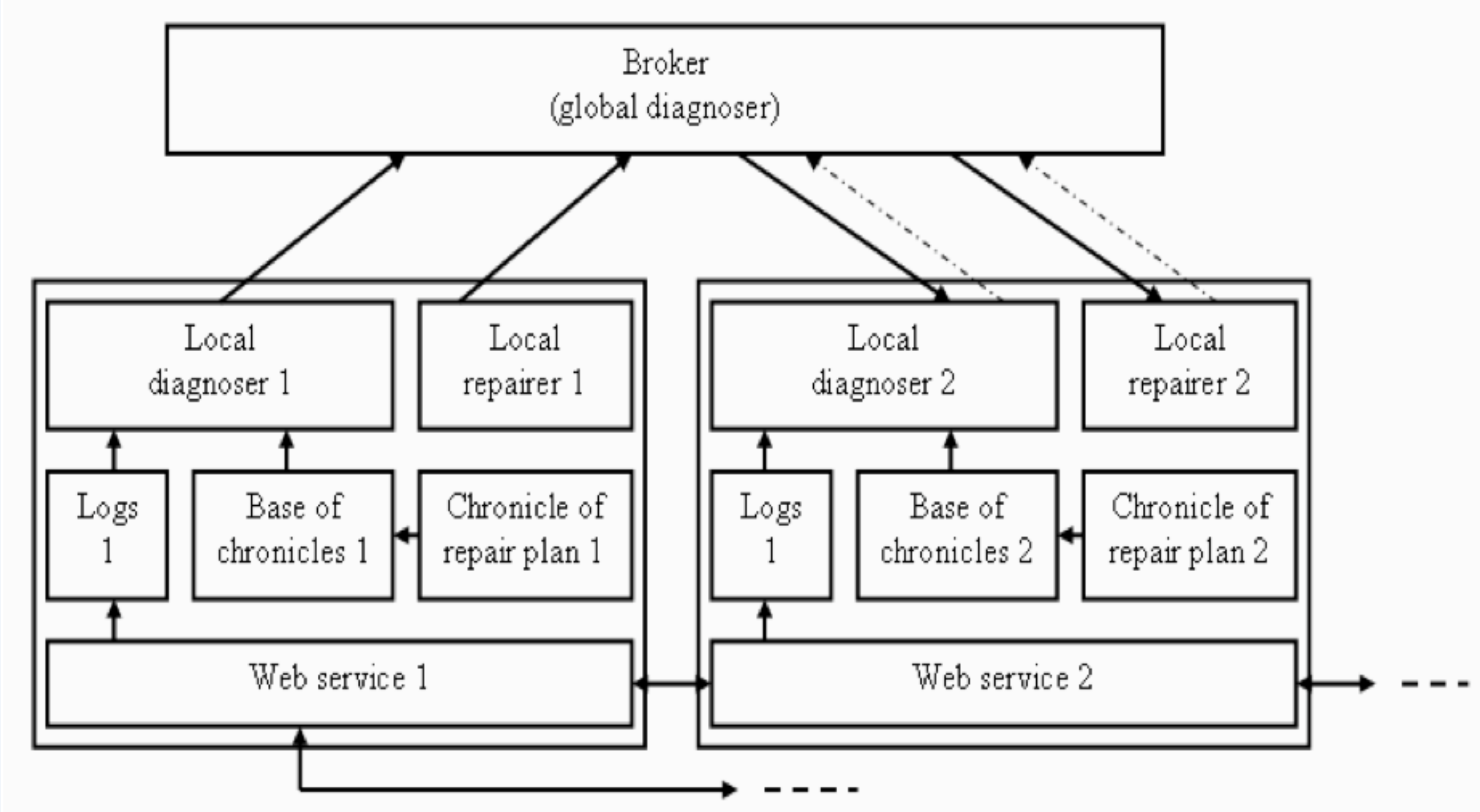
requestShipping

Le cadre applicatif

- Distribué ⇒ infection = propagation des fautes



Architecture



Réparer ?

Réparer ?

- Plan de réparation ?

Réparer ?

- Plan de réparation ?

  local = ensemble d’actions élémentaires :

  - Substitute(WS1,WS2) + Redo(a2)
  - Compensate(a4,a3,a2) + Retry(a2)
  - Compensate(a4,a3,a2) + ChgValue(code,a2) + Redo(a2)

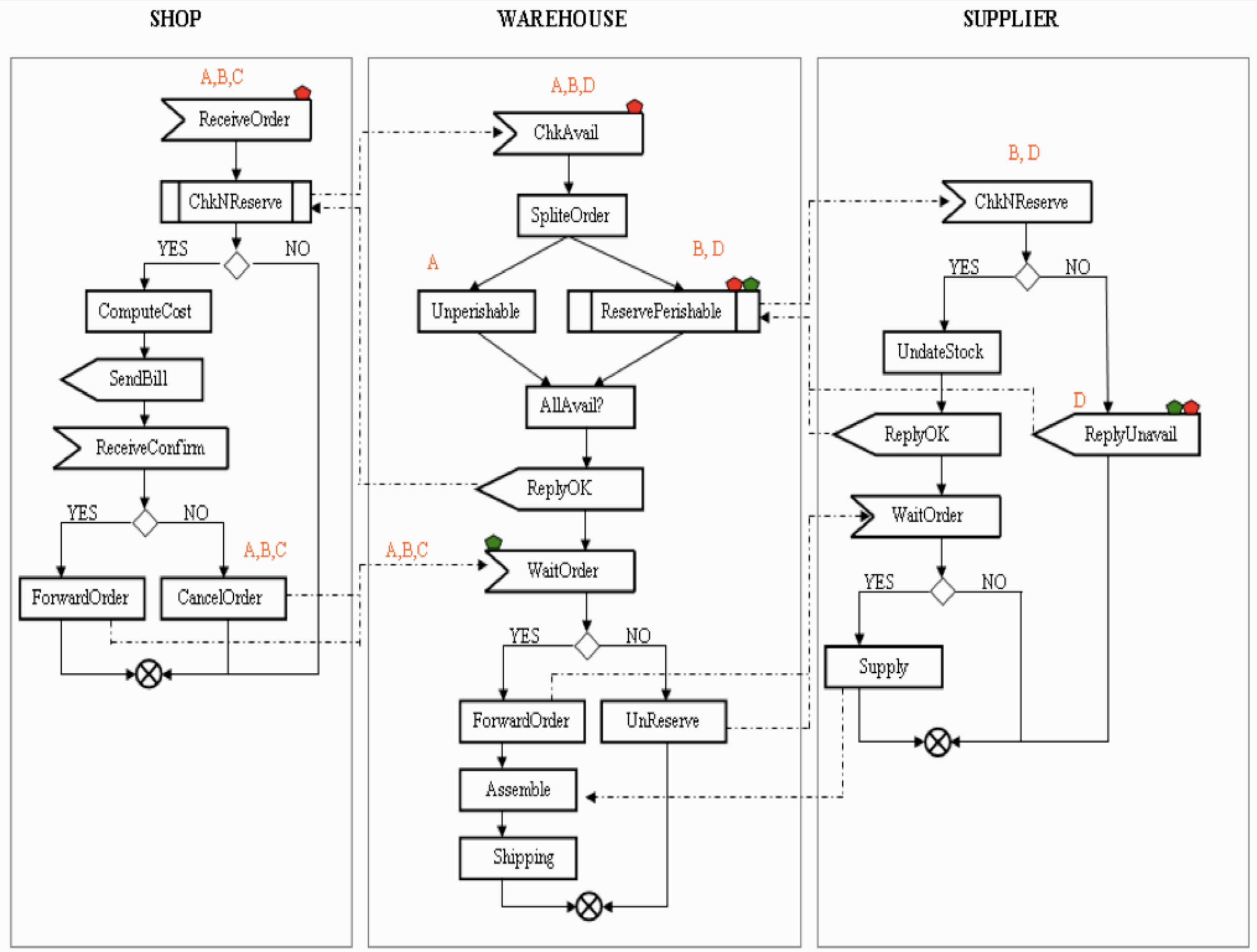
Réparer ?

- Plan de réparation ?

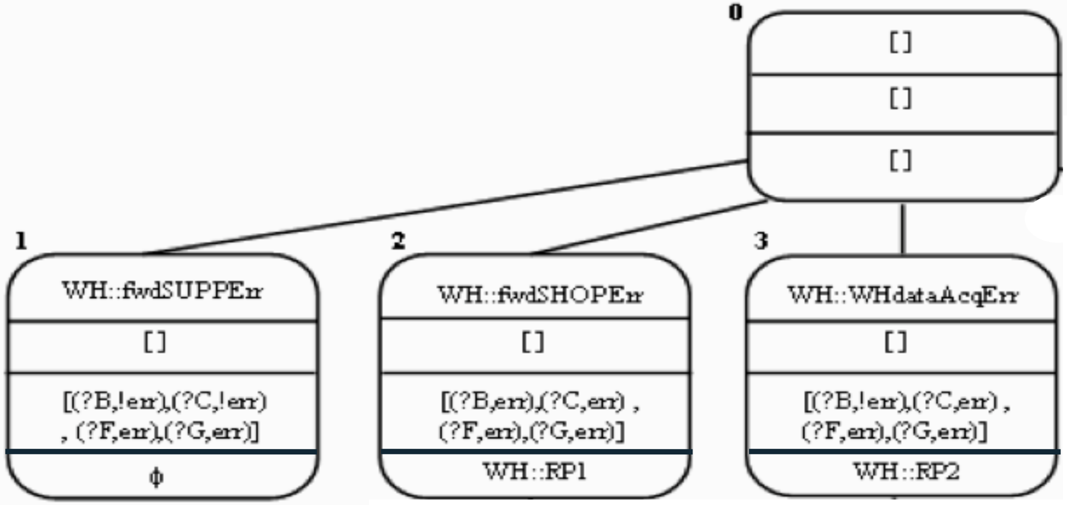
  local = ensemble d’actions élémentaires :
  - Substitute(WS1,WS2) + Redo(a2)
  - Compensate(a4,a3,a2) + Retry(a2)
  - Compensate(a4,a3,a2) + ChgValue(code,a2) + Redo(a2)

- 3 phases :
  - Stopper le processus
  - Roll-back (compensation)
  - Relancer (ou se mettre en attente…)

Exemple


SHOP
WAREHOUSE
SUPPLIER
A,B,C
ReceiveOrder
ChkNReserve
YES
NO
ComputeCost
SendBill
ReceiveConfirm
YES
NO
A,B,C
ForwardOrder
CancelOrder
A,B,D
ChkAvail
SpliteOrder
A
B, D
Unperishable
ReservePerishable
AllAvail?
ReplyOK
A,B,C
WaitOrder
YES
NO
ForwardOrder
UnReserve
Assemble
Shipping
B, D
ChkNReserve
YES
NO
UndateStock
D
ReplyOK
ReplyUnavail
WaitOrder
YES
NO
Supply

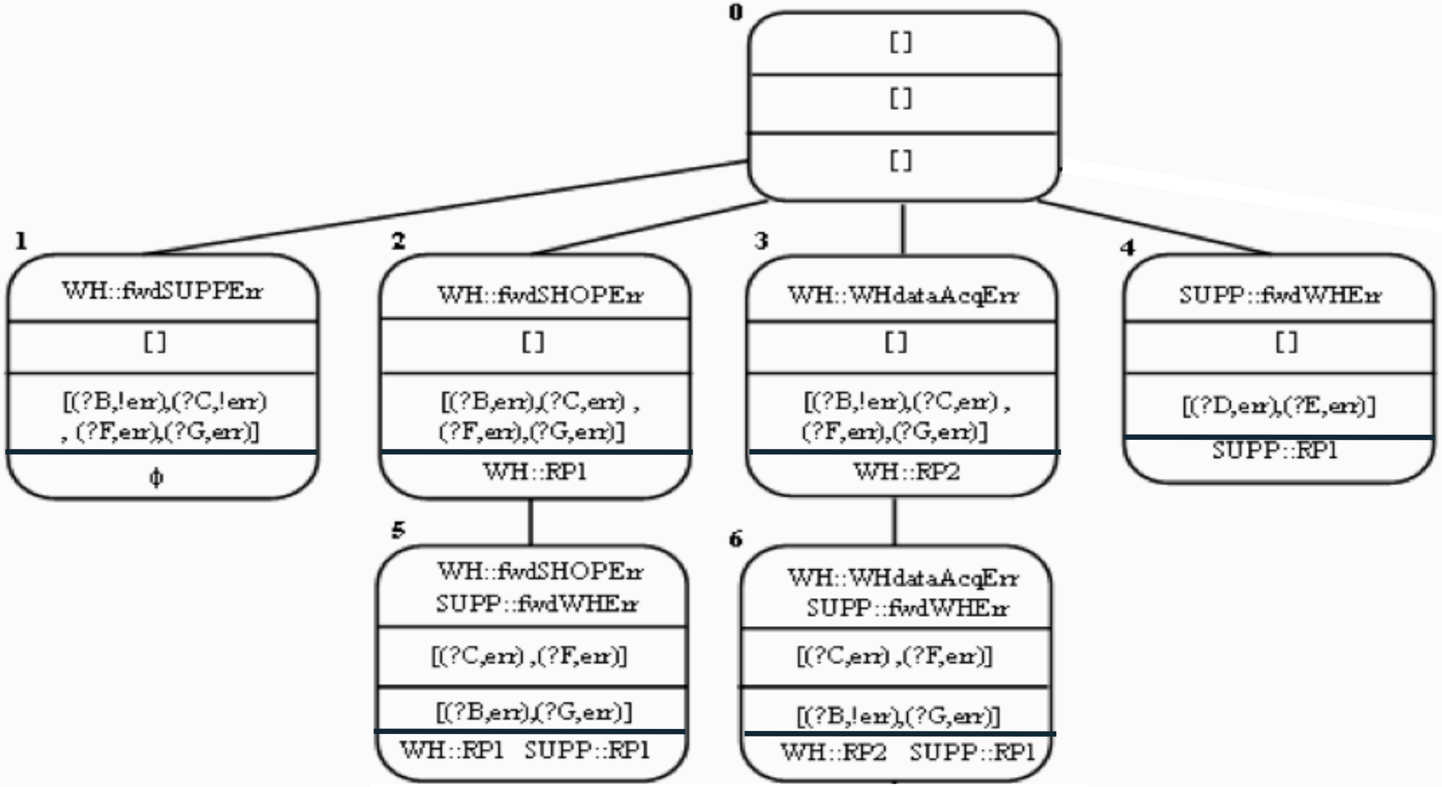
Arbre de diag/répar



A: ?ShopListOut
B: ?WHListIn
C: ?WHItemsOut
D: ?SuppItemsIn
E: ?SuppItemsOut
F: ?WHItemsIn
G: ?WHListOut
H: ?ShopListIn

A=B , C=D, E=F, G=H

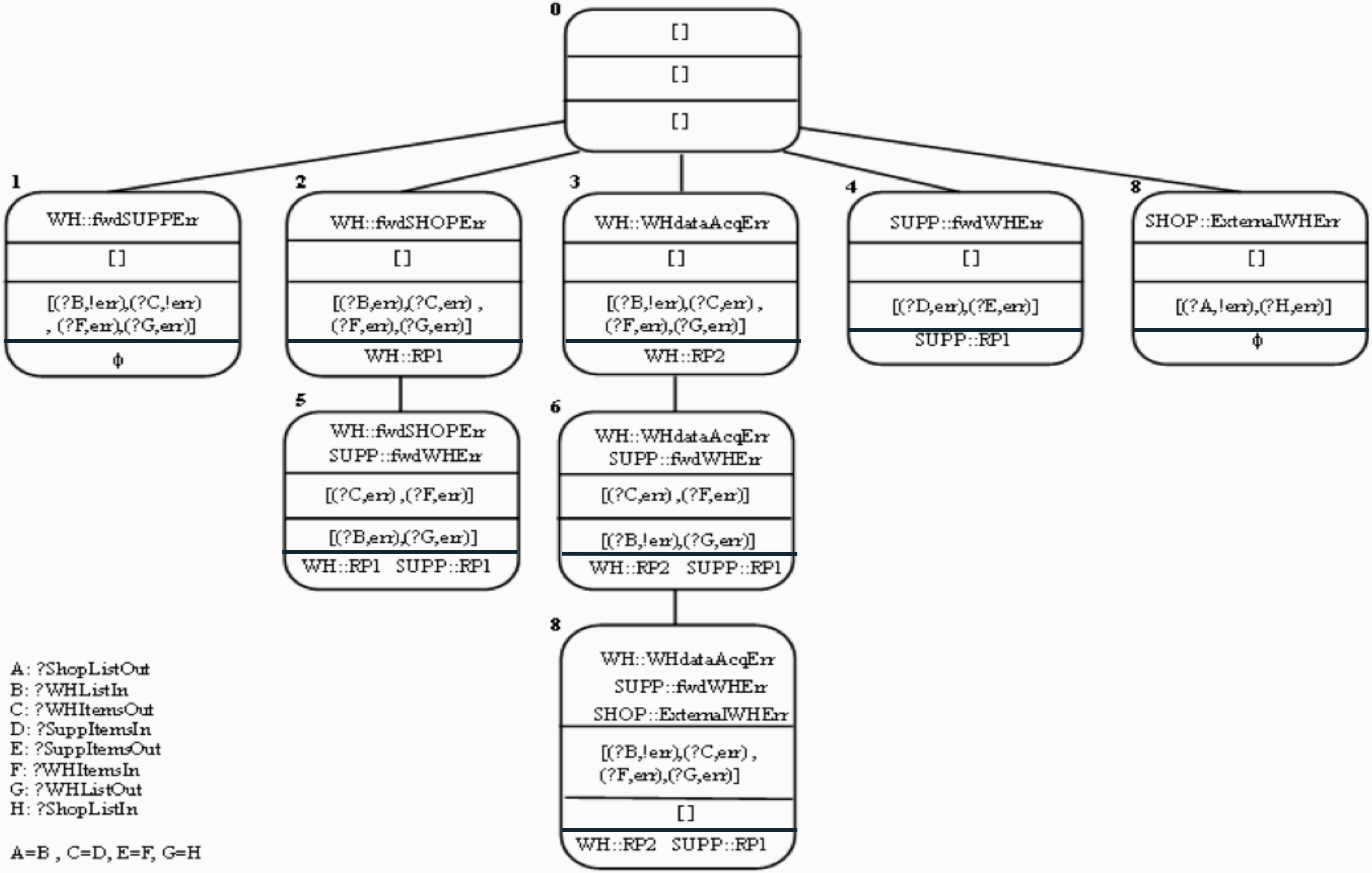
Nouvel arbre de diag/ répar



A: ?ShopListOut
B: ?WHListIn
C: ?WHItemsOut
D: ?SuppItemsIn
E: ?SuppItemsOut
F: ?WHItemsIn
G: ?WHListOut
H: ?ShopListIn

A=B , C=D, E=F, G=H

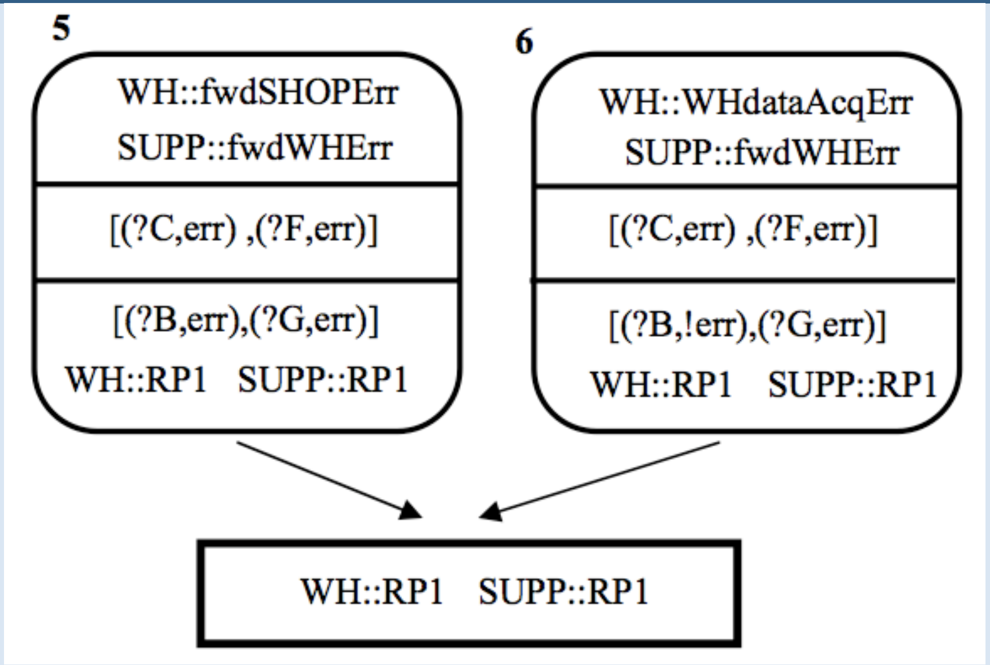
Nouvel arbre de diag/ répar



A: ?ShopListOut
B: ?WHListIn
C: ?WHItemsOut
D: ?SuppItemsIn
E: ?SuppItemsOut
F: ?WHItemsIn
G: ?WHListOut
H: ?ShopListIn

A=B , C=D, E=F, G=H

Cas d’un plan commun



5

WH::fwdSHOPErr
SUPP::fwdWHErr

[(?C,err) ,(?F,err)]

[(?B,err),(?G,err)]

WH::RP1 SUPP::RP1

6

WH::WHdataAcqErr
SUPP::fwdWHErr

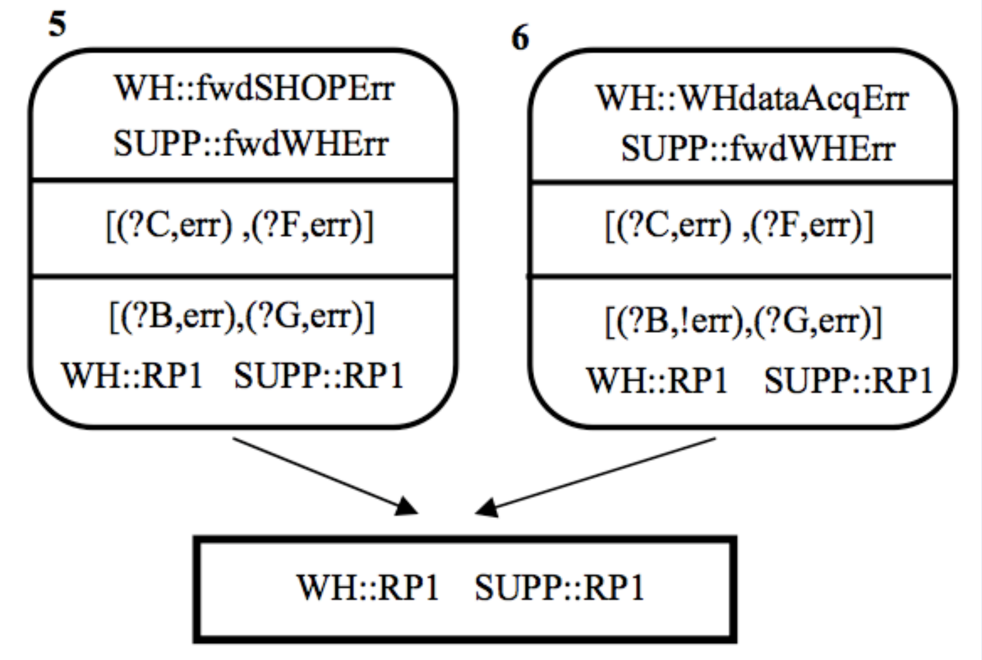
[(?C,err) ,(?F,err)]

[(?B,!err),(?G,err)]

WH::RP1 SUPP::RP1

WH::RP1 SUPP::RP1

Cas d’un plan commun



- Plus complexe : cas où un plan RP1 « est inclus » dans RP2 (RP2 « subsume » RP1)
  - RP2 peut remplacer RP1 -> plan commun

Extensions et prospectives

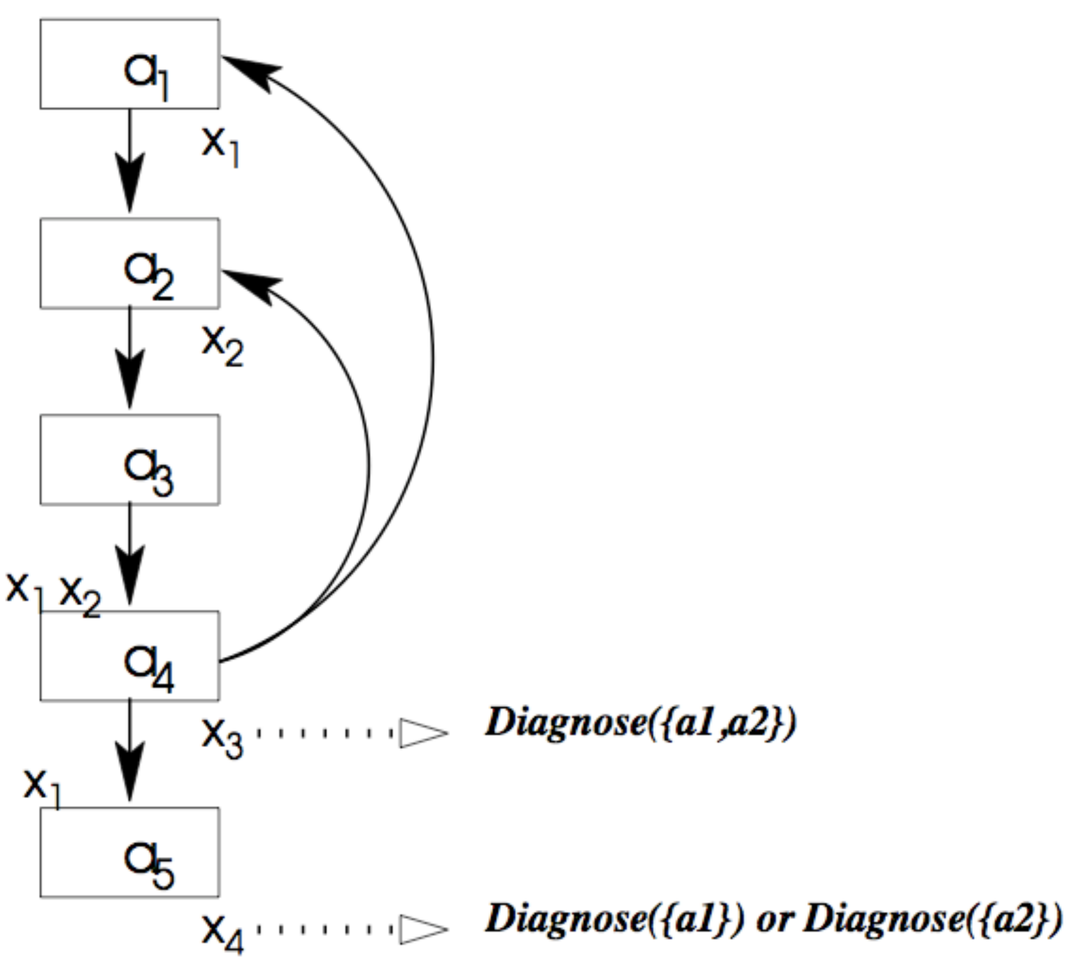
Prise en compte des délais

- Hypothèse non confirmée mais plan urgent = plus compensable au-delà

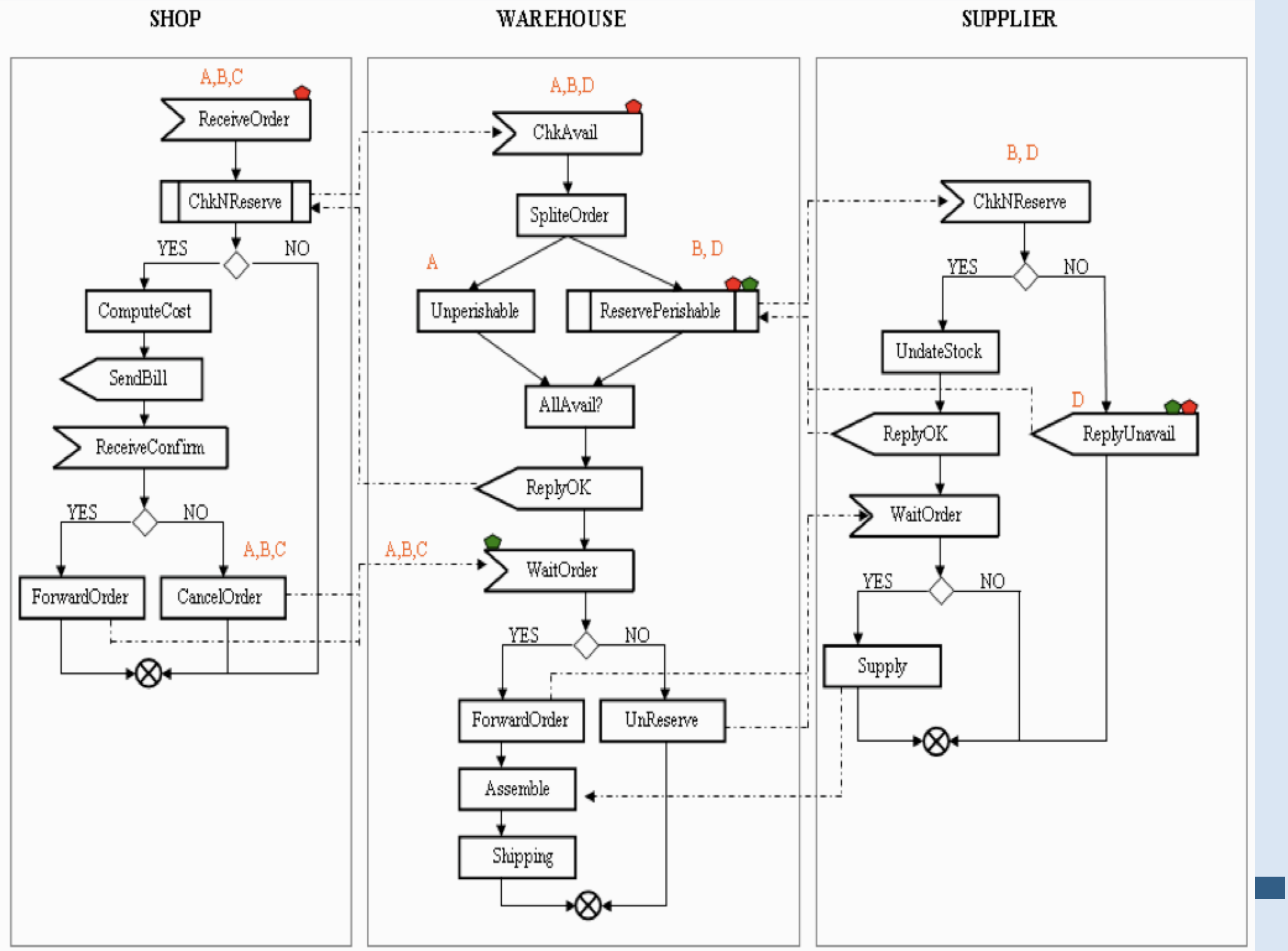
  => contraintes temporelles :

  Diag-min-delay(F) ≤ Repair-max-delay(F)

Cadre web-services distribués



Exemple


SHOP
WAREHOUSE
SUPPLIER
A,B,C
ReceiveOrder
ChkNReserve
YES
NO
ComputeCost
SendBill
ReceiveConfirm
YES
NO
A,B,C
ForwardOrder
CancelOrder
A,B,D
ChkAvail
SpliteOrder
A
B, D
Unperishable
ReservePerishable
AllAvail?
ReplyOK
A,B,C
WaitOrder
YES
NO
ForwardOrder
UnReserve
Assemble
Shipping
B, D
ChkNReserve
YES
NO
UndateStock
D
ReplyOK
ReplyUnavail
WaitOrder
YES
NO
Supply

Cadre décisionnel plus général

Cadre décisionnel plus général

- Le système évolue dans un graphe G=(V,E)

Cadre décisionnel plus général

- Le système évolue dans un graphe G=(V,E)
    - Nœuds V = <F_j, R_j>

Cadre décisionnel plus général

- Le système évolue dans un graphe G=(V,E)
  - Nœuds V = <F_j, R_j>
    - R_j : macro-plan = ensemble des plans r_k tels que *Repair*(r_k, F_j) + plan vide ∅

Cadre décisionnel plus général

- Le système évolue dans un graphe G=(V,E)
  - Nœuds V = $<F_j, R_j>$
    - R_j : macro-plan = ensemble des plans r_k tels que *Repair*(r_k, F_j) + plan vide ∅

Cadre décisionnel plus général

- Le système évolue dans un graphe G=(V,E)
  - Nœuds V = <F_j, R_j>
    - R_j : macro-plan = ensemble des plans r_k tels que *Repair(r_k, F_j)* + plan vide ∅
  - Transitions E = application d'un des r_k de R_j

Divers cas de figure

Divers cas de figure

- Quatre possibilités de macro-fautes

Divers cas de figure

- Quatre possibilités de macro-fautes

Divers cas de figure

- Quatre possibilités de macro-fautes
  - F={ok} => macro-plan R={∅}

Divers cas de figure

- Quatre possibilités de macro-fautes
  - F={ok} => macro-plan R={∅}
  - F={f_1} => diag discriminant
    - on appliquera généralement un plan s'il existe

Divers cas de figure

- Quatre possibilités de macro-fautes
  - $F=\{ok\}$ => macro-plan $R=\{\emptyset\}$
  - $F=\{f_1\}$ => diag discriminant
    - on appliquera généralement un plan s’il existe
  - $F=\{f_1,f_2…\}$ => diag non discriminant

Divers cas de figure

- Quatre possibilités de macro-fautes
  - $F=\{ok\}$ => macro-plan $R=\{\varnothing\}$
  - $F=\{f_1\}$ => diag discriminant
    - on appliquera généralement un plan s’il existe
  - $F=\{f_1,f_2\ldots\}$ => diag non discriminant
    - appliquer un plan ou attendre de discriminer ?...

Divers cas de figure

- Quatre possibilités de macro-fautes
  - $F=\{ok\}$ => macro-plan $R=\{\varnothing\}$
  - $F=\{f_1\}$ => diag discriminant
    - on appliquera généralement un plan s’il existe
  - $F=\{f_1,f_2\ldots\}$ => diag non discriminant
    - appliquer un plan ou attendre de discriminer ?...

Divers cas de figure

- Quatre possibilités de macro-fautes
  - F={ok} => macro-plan R={∅}
  - F={f_1} => diag discriminant
    - on appliquera généralement un plan s'il existe
  - F={f_1,f_2…} => diag non discriminant
    - appliquer un plan ou attendre de discriminer ?…
  - F={ok,f_1…} => indécision…

Divers cas de figure

- Quatre possibilités de macro-fautes
  - $F=\{ok\}$ => macro-plan $R=\{\varnothing\}$
  - $F=\{f_1\}$ => diag discriminant
    - on appliquera généralement un plan s'il existe
  - $F=\{f_1,f_2\ldots\}$ => diag non discriminant
    - appliquer un plan ou attendre de discriminer ?...
  - $F=\{ok,f_1\ldots\}$ => indécision...
    - attendre ou appliquer quand même un plan potentiellement inutile...?

Plans dynamiques

Plans dynamiques

- Plans pré-existants : peu réaliste...

Plans dynamiques

- Plans pré-existants : peu réaliste…

Plans dynamiques

- Plans pré-existants : peu réaliste…
- Remplacer par la notion de stratégie applicable

Plans dynamiques

- Plans pré-existants : peu réaliste…
- Remplacer par la notion de stratégie applicable
  - Stratégie = goal + actions imposées → plan possible (à construire en ligne)
  - Stratégie = contraintes nouvelles → plan = rétablir la cohérence
  - etc

Plans dynamiques

- Plans pré-existants : peu réaliste…
- Remplacer par la notion de stratégie applicable
  - Stratégie = goal + actions imposées → plan possible (à construire en ligne)
  - Stratégie = contraintes nouvelles → plan = rétablir la cohérence
  - etc

Plans dynamiques

- Plans pré-existants : peu réaliste...

- Remplacer par la notion de stratégie applicable
  - Stratégie = goal + actions imposées → plan possible (à construire en ligne)
  - Stratégie = contraintes nouvelles → plan = rétablir la cohérence
  - etc

- Nécessite d’intégrer des concepts de planification / CSPs...

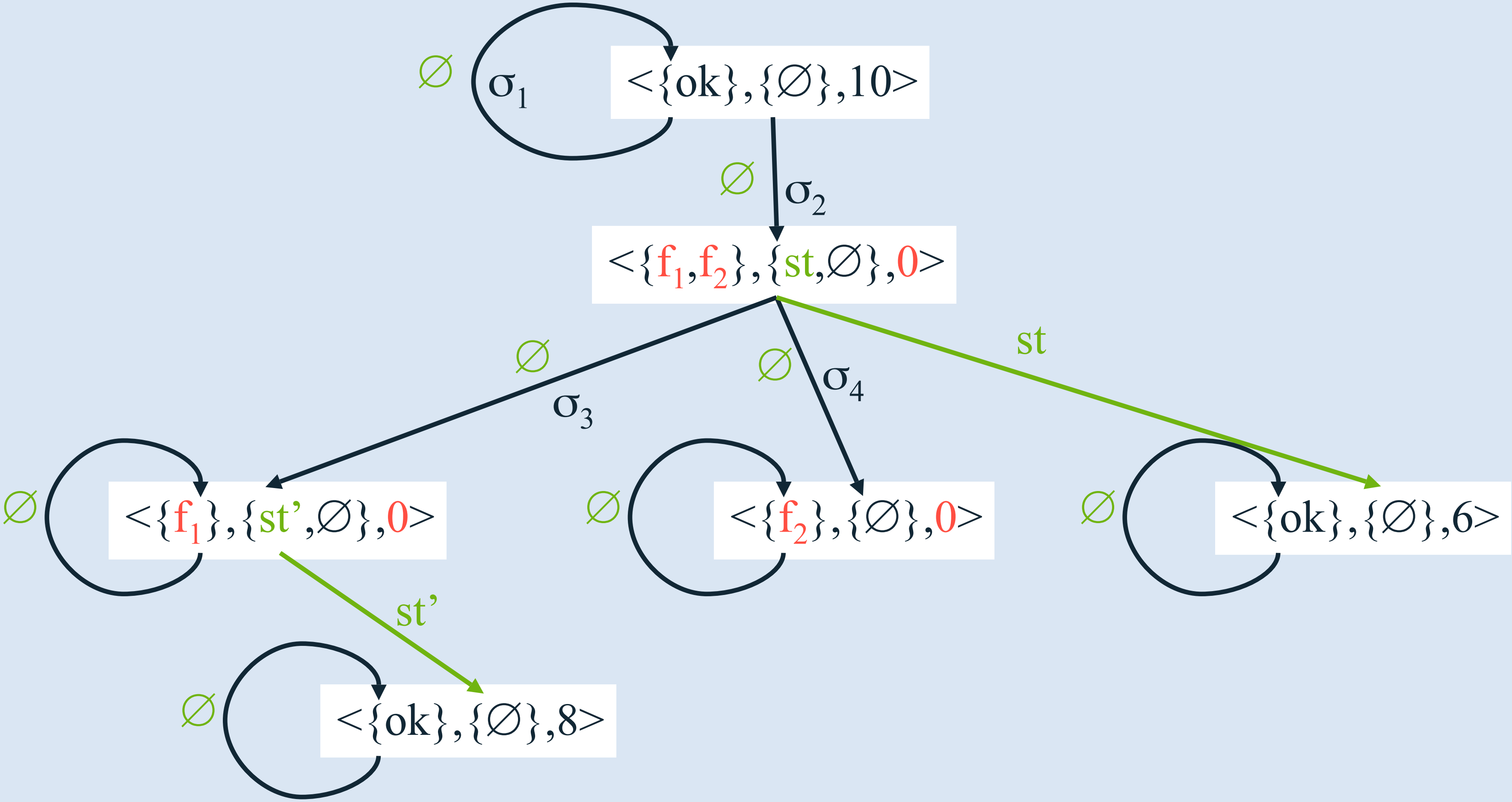
Plans dynamiques

- Plans pré-existants : peu réaliste…
- Remplacer par la notion de stratégie applicable
  - Stratégie = goal + actions imposées → plan possible (à construire en ligne)
  - Stratégie = contraintes nouvelles → plan = rétablir la cohérence
  - etc
- Nécessite d'intégrer des concepts de planification / CSPs…

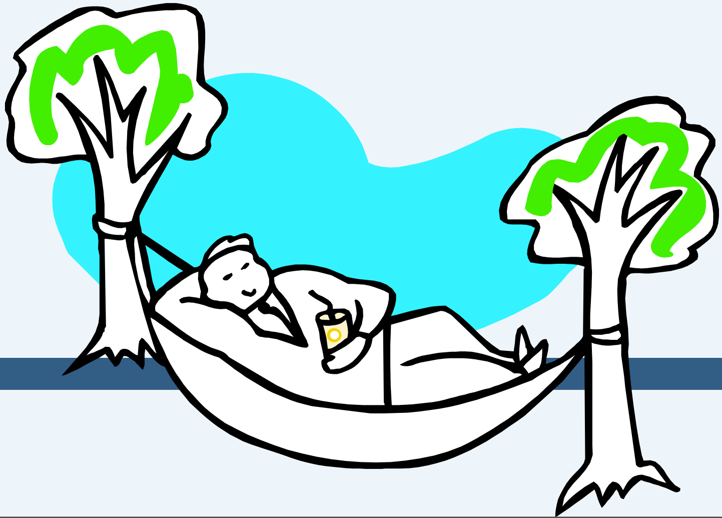
Plans dynamiques

- Plans pré-existants : peu réaliste…
- Remplacer par la notion de stratégie applicable
  - Stratégie = goal + actions imposées → plan possible (à construire en ligne)
  - Stratégie = contraintes nouvelles → plan = rétablir la cohérence
  - etc
- Nécessite d’intégrer des concepts de planification / CSPs…
- Notions de qualité / optimisation…

Processus décisionnel



Merci !



Questions?

thierry.vidal@enit.fr