Distributed diagnosis of concurrent and asynchronous Discrete Event Systems

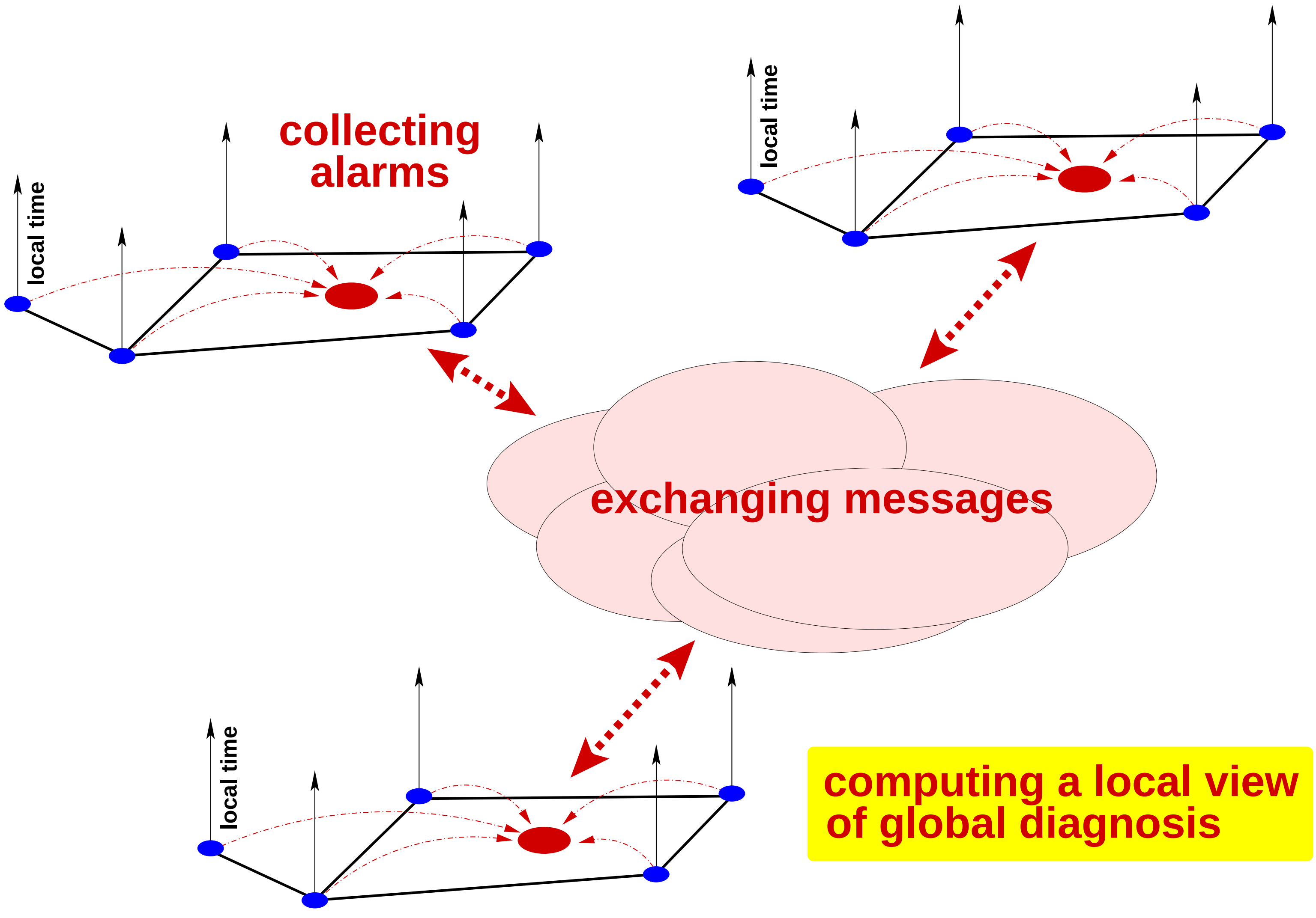
Albert Benveniste

Eric Fabre, Stefan Haar, Claude Jard

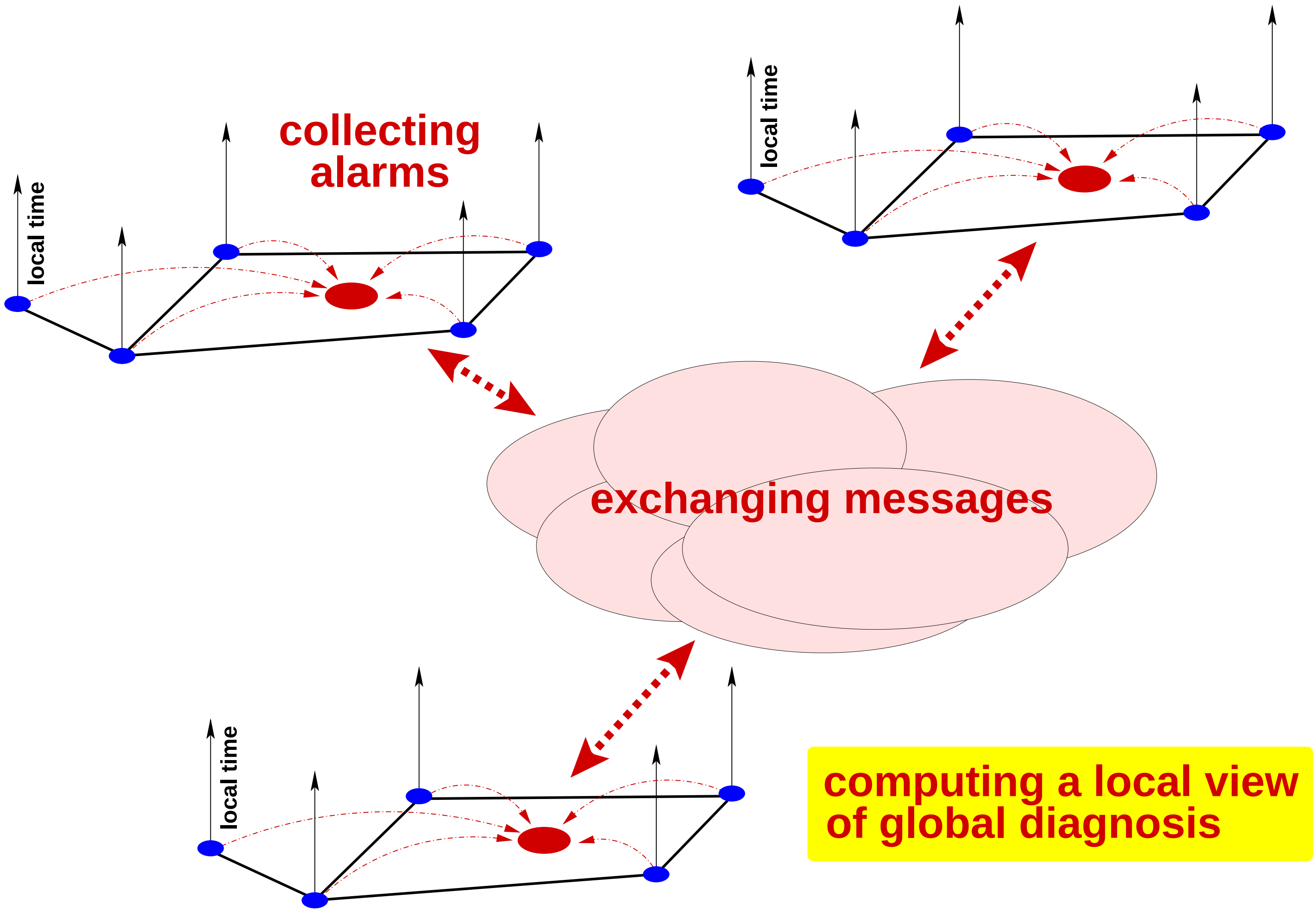
IRISA, Rennes

2004

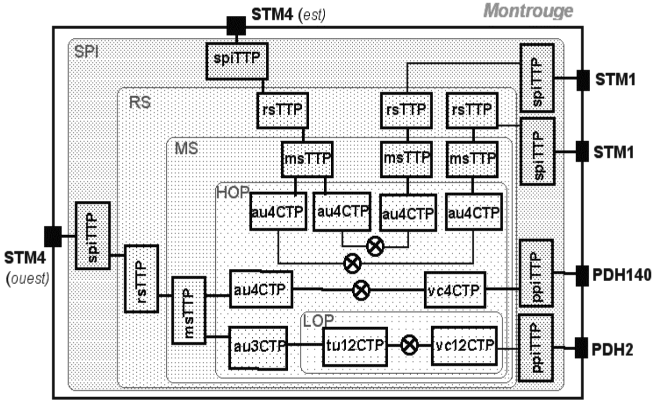
The problem:

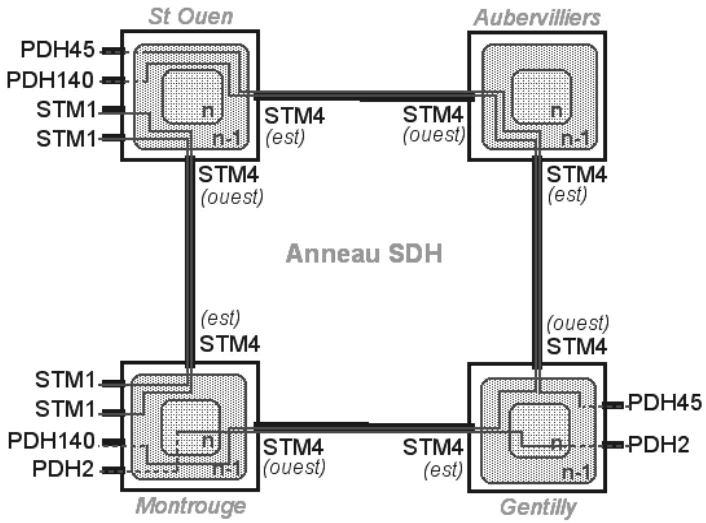

collecting
alarms
local time
local time
local time
exchanging messages
computing a local view
of global diagnosis

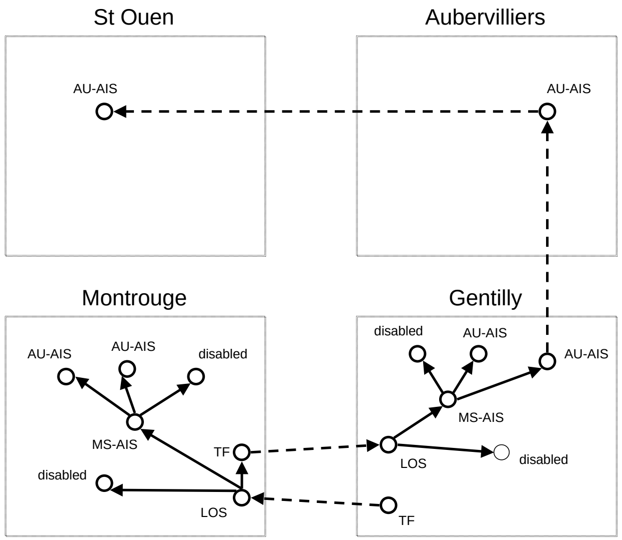
The problem: distributed + asynchronous



The SDH/SONET ring in the Paris area







Features

- **distributed algorithm**
  - *synchronization services should not be used*
  - *some reliability can be assumed (error correcting codes)*
- **nontrivial even if not distributed**
  - *recover hidden state history from observation sequence*
  - *ambiguities* $\Rightarrow$ *nondeterminism, probabilistic*

1. A toy example:

   Petri nets and unfoldings

   asynchronous diagnosis

   distributed diagnosis

2. Formalizing: Petri nets, unfoldings and event structures

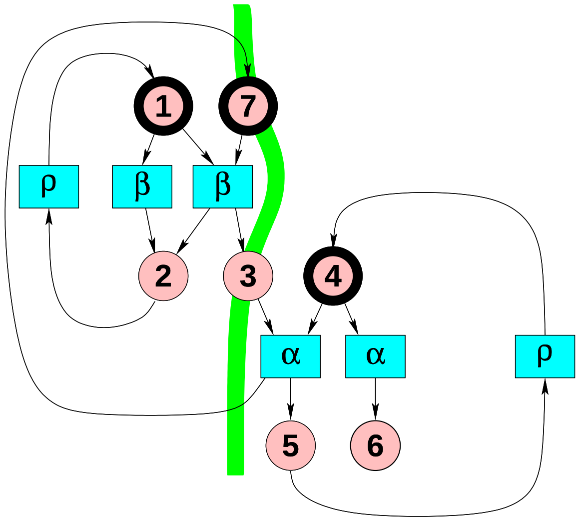
3. An abstract setting

4. Distributed orchestration:

   tree-shaped networks

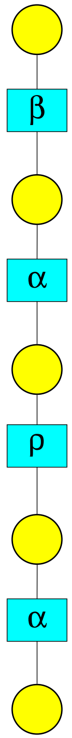
   general networks

A toy example

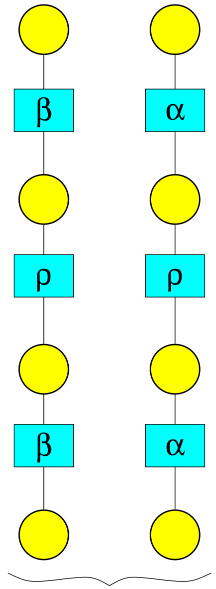


1,4: safe states
2,6: faulty states
5: faulty by interaction

Petri net:
global, or
2 components

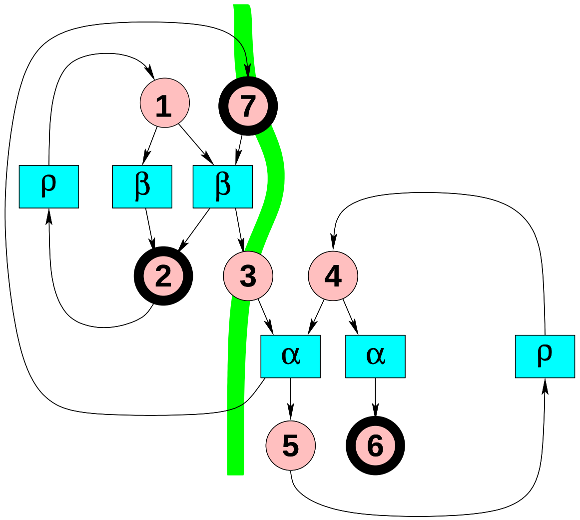


alarms:
1 sensor



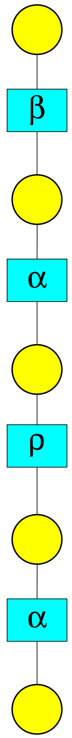
alarms:
2 independent sensors

A toy example

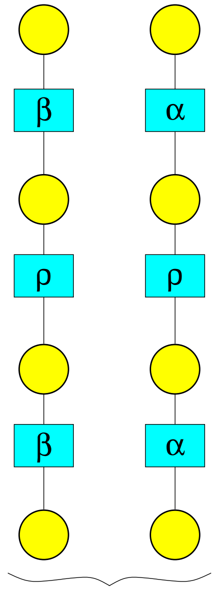


1,4: safe states
2,6: faulty states
5: faulty by interaction

Petri net:
global, or
2 components

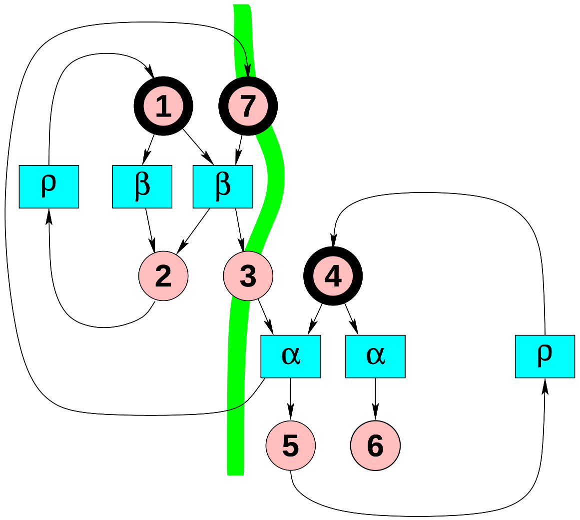


alarms:
1 sensor



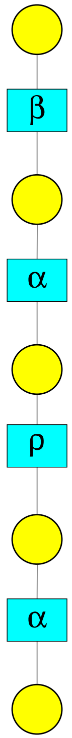
alarms:
2 independent
sensors

A toy example

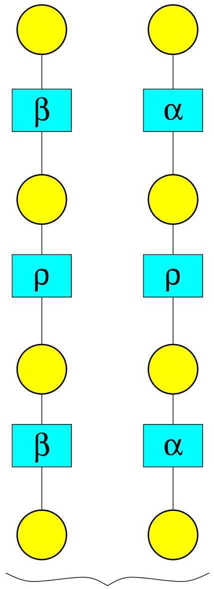


1,4: safe states
2,6: faulty states
5: faulty by interaction

Petri net:
global, or
2 components

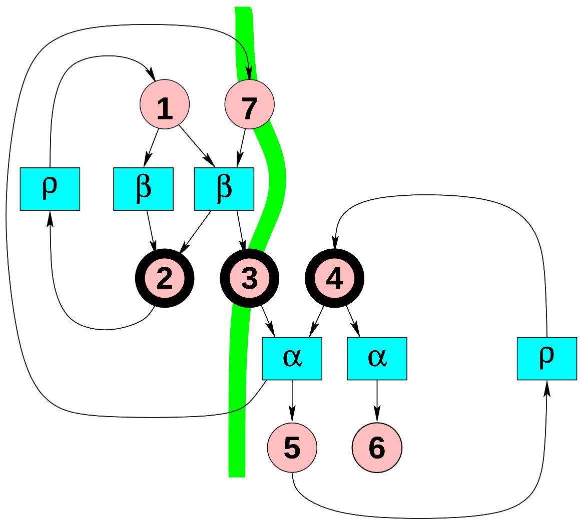


alarms:
1 sensor



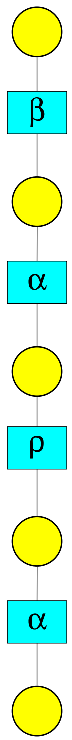
alarms:
2 independent
sensors

A toy example

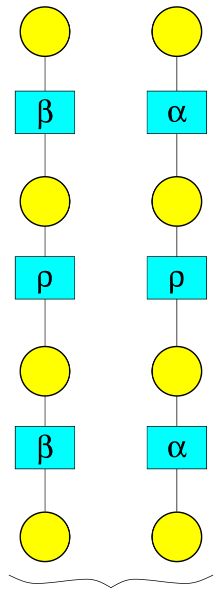


1,4: safe states
2,6: faulty states
5: faulty by interaction

Petri net:
global, or
2 components

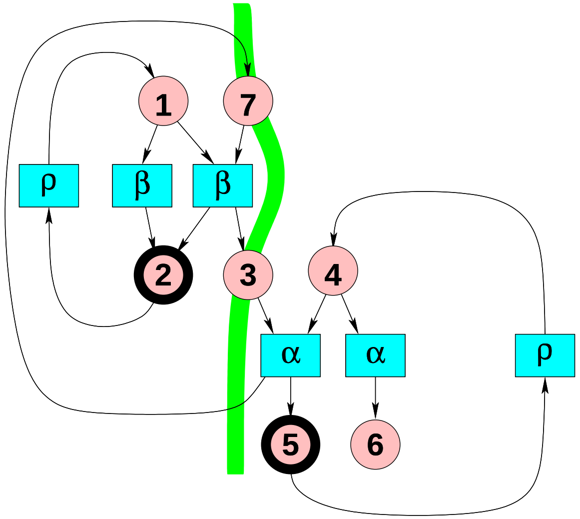


alarms:
1 sensor



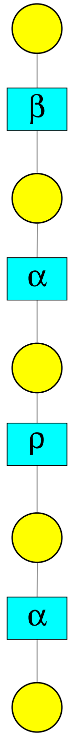
alarms:
2 independent
sensors

A toy example

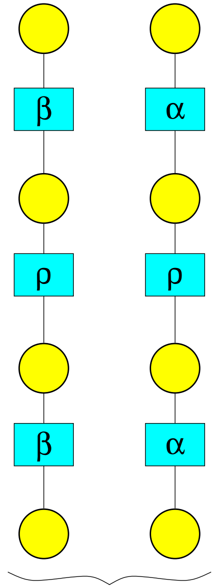


1,4: safe states
2,6: faulty states
5: faulty by interaction

Petri net:
global, or
2 components

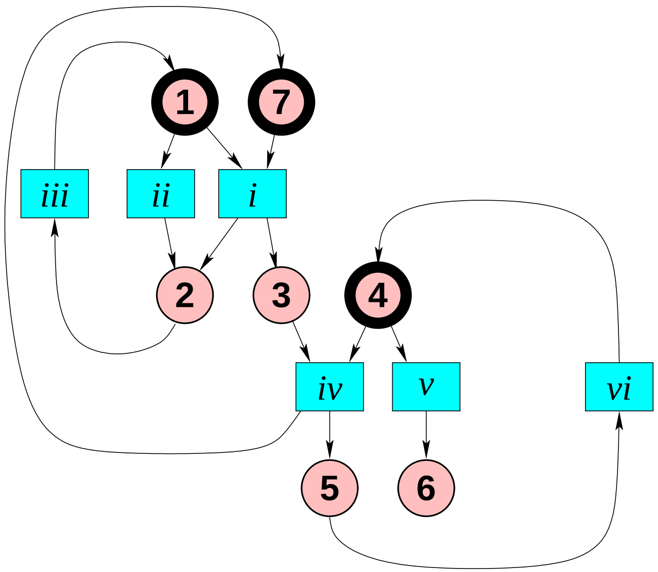


alarms:
1 sensor



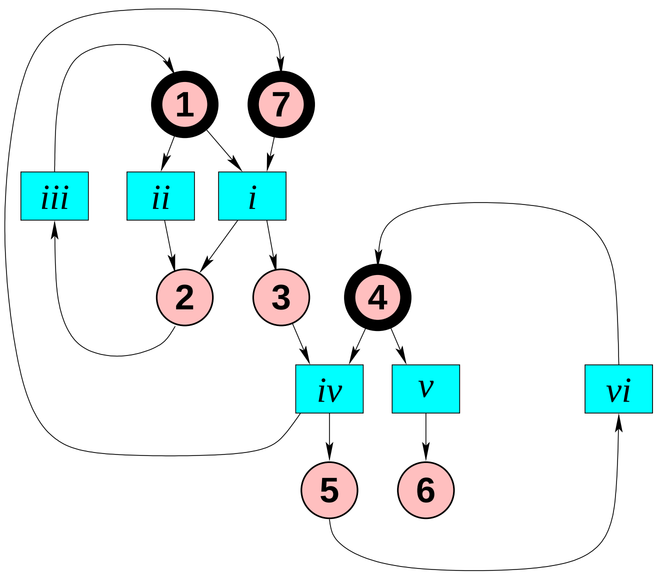
alarms:
2 independent
sensors

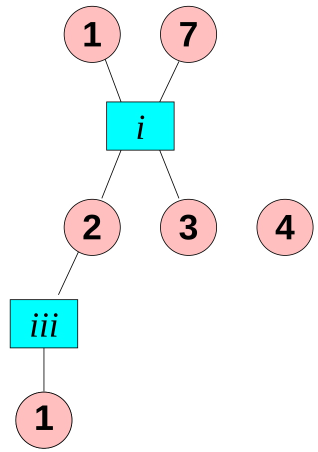
Unfoldings: $\mathcal{P}$



a structure to represent sets of traces with concurrency

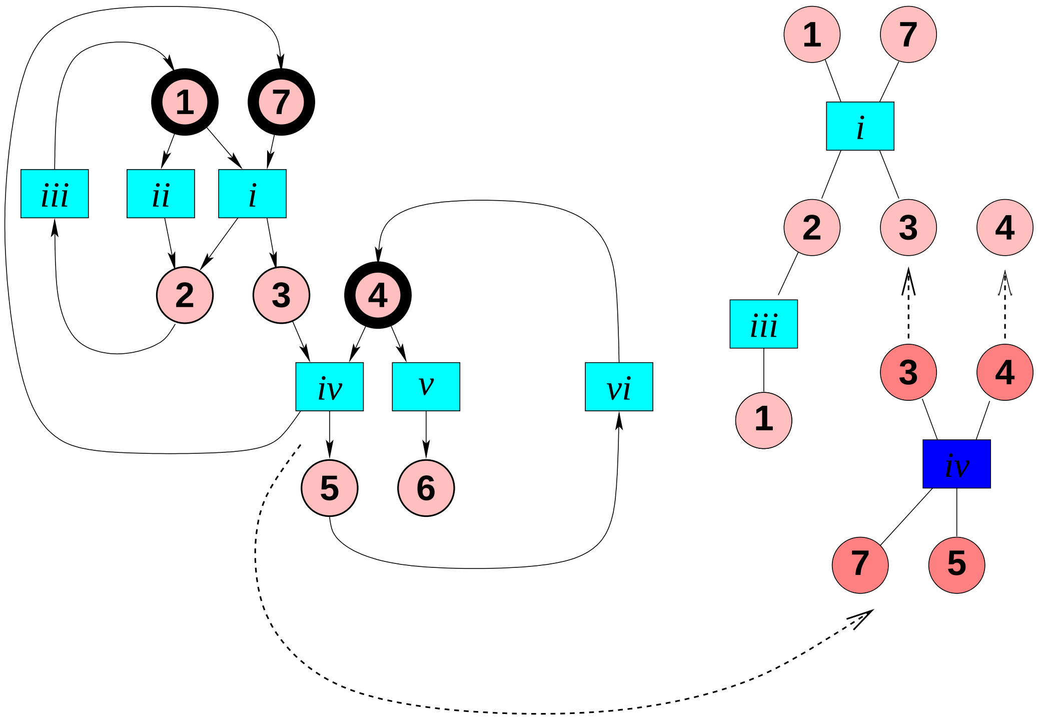
Unfoldings: $\mathcal{P}$





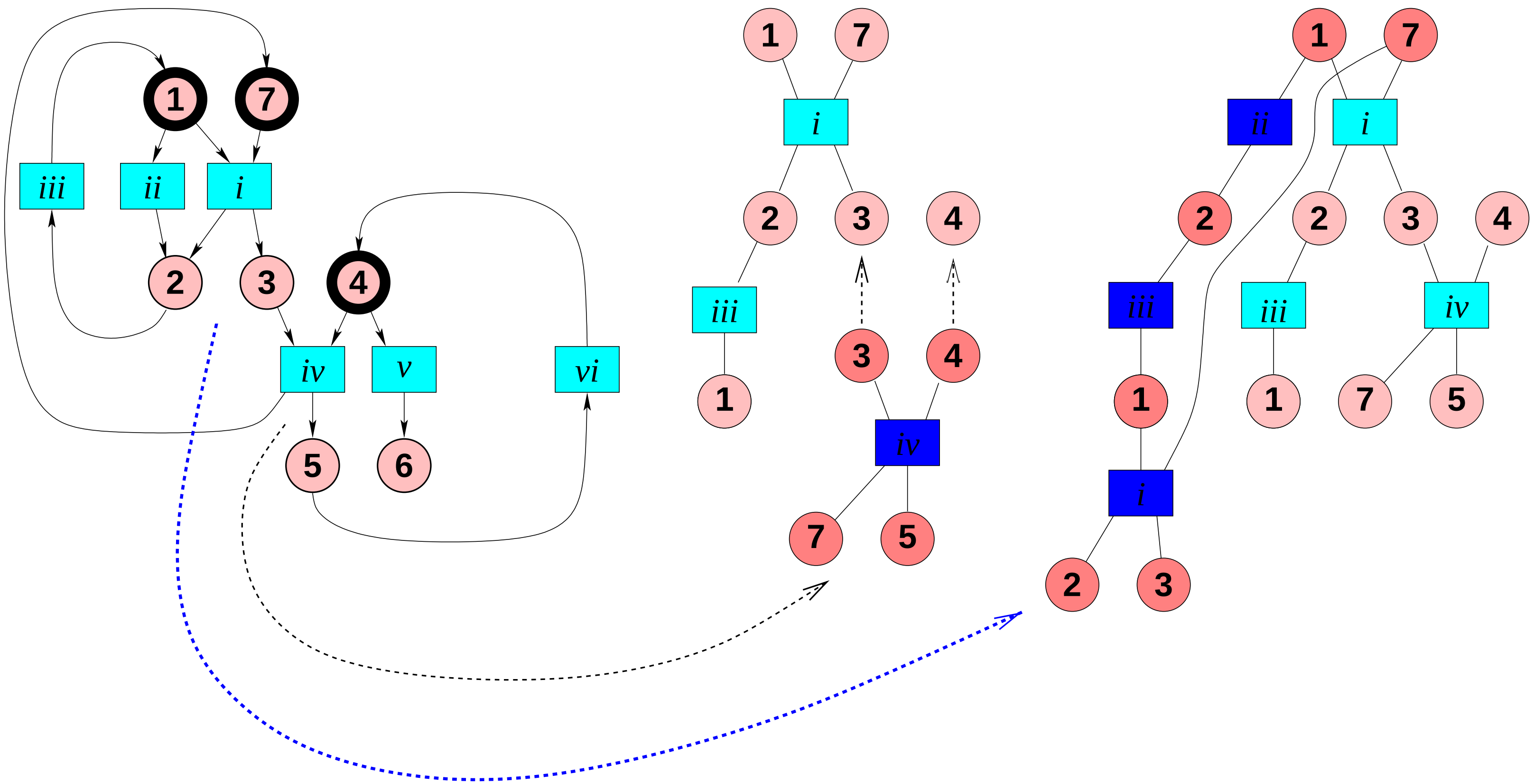
a structure to represent sets of traces with concurrency

Unfoldings: $\mathcal{P}$, $\mathcal{U}_{\mathcal{P}}$



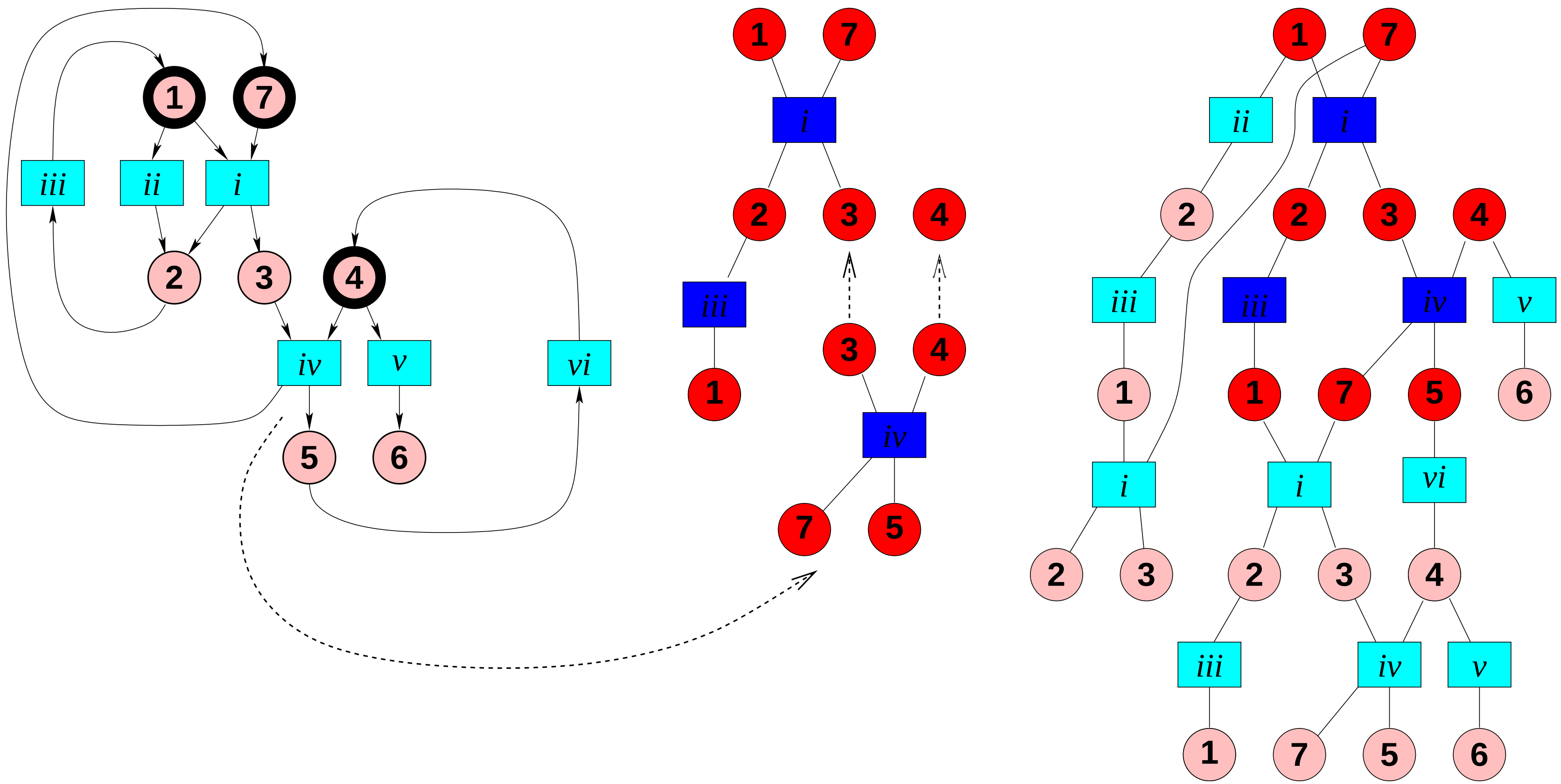
a structure to represent sets of traces with concurrency

Unfoldings: $\mathcal{P}$, $\mathcal{U}_{\mathcal{P}}$



a structure to represent sets of traces with concurrency

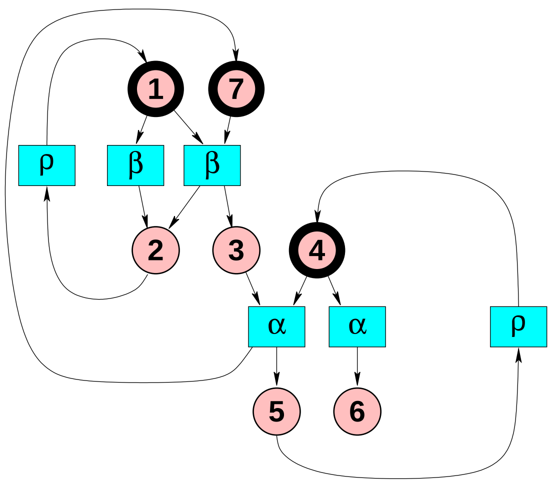
Unfoldings: $\mathcal{P}$, $\mathcal{U}_{\mathcal{P}}$

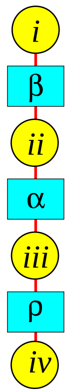


a structure to represent sets of traces with concurrency

1. A toy example:
   - Petri nets and unfoldings
   - asynchronous diagnosis
   - distributed diagnosis
2. Formalizing: Petri nets, unfoldings and event structures
3. An abstract setting
4. Distributed orchestration:
   - tree-shaped networks
   - general networks

Diagnets: $\mathcal{P}, \mathcal{A}$

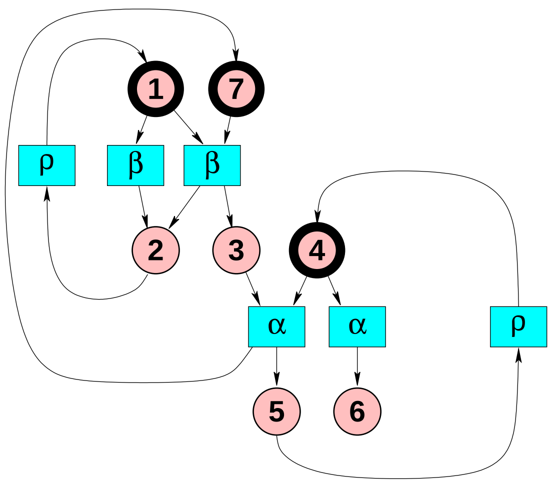




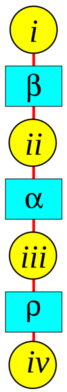
net

alarm net

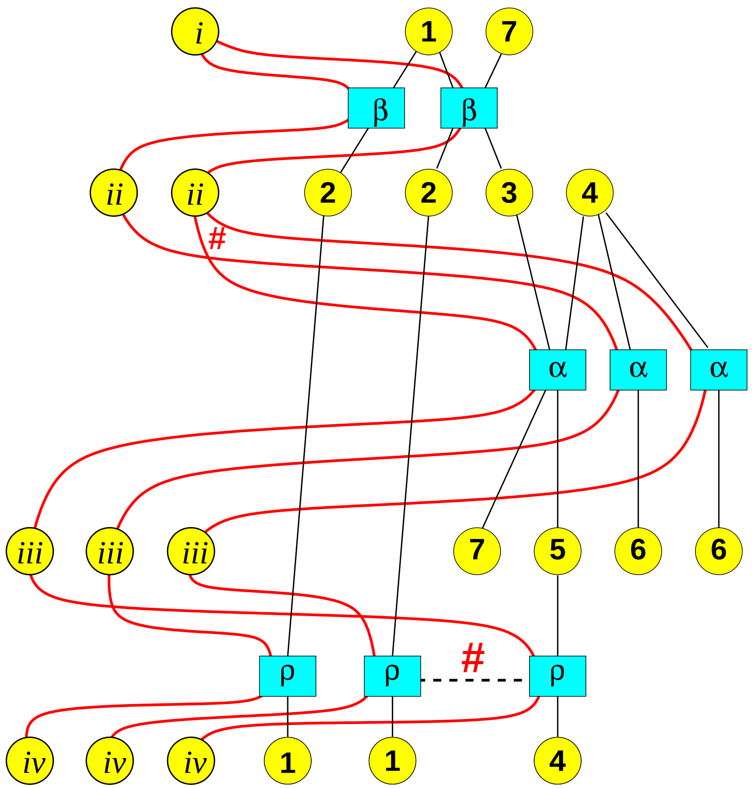
Diagnets: $\mathcal{P}, \mathcal{A}$, $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$



net

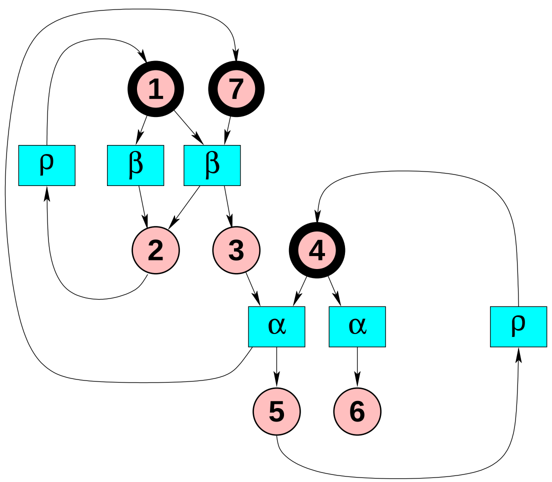


alarm net

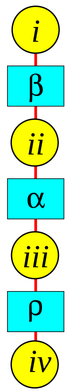


diagnet

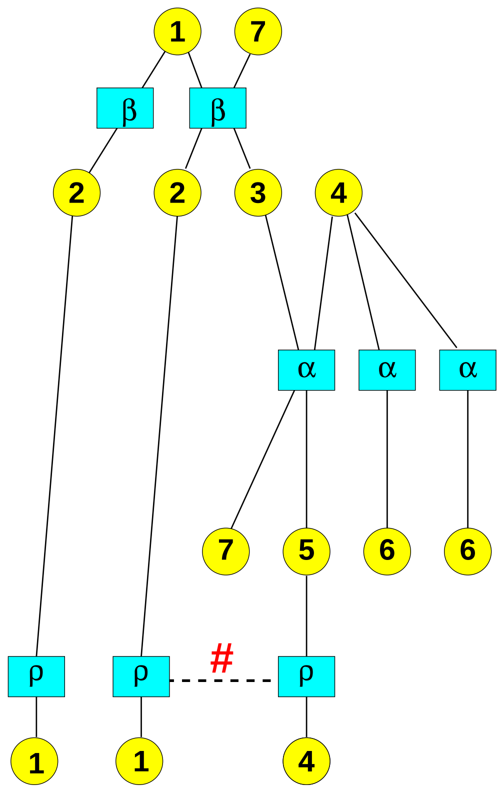
Diagnets: $\mathcal{P}, \mathcal{A}$, $\pi_P \left(\mathcal{U}_{\mathcal{P} \times \mathcal{A}} \right)$



net

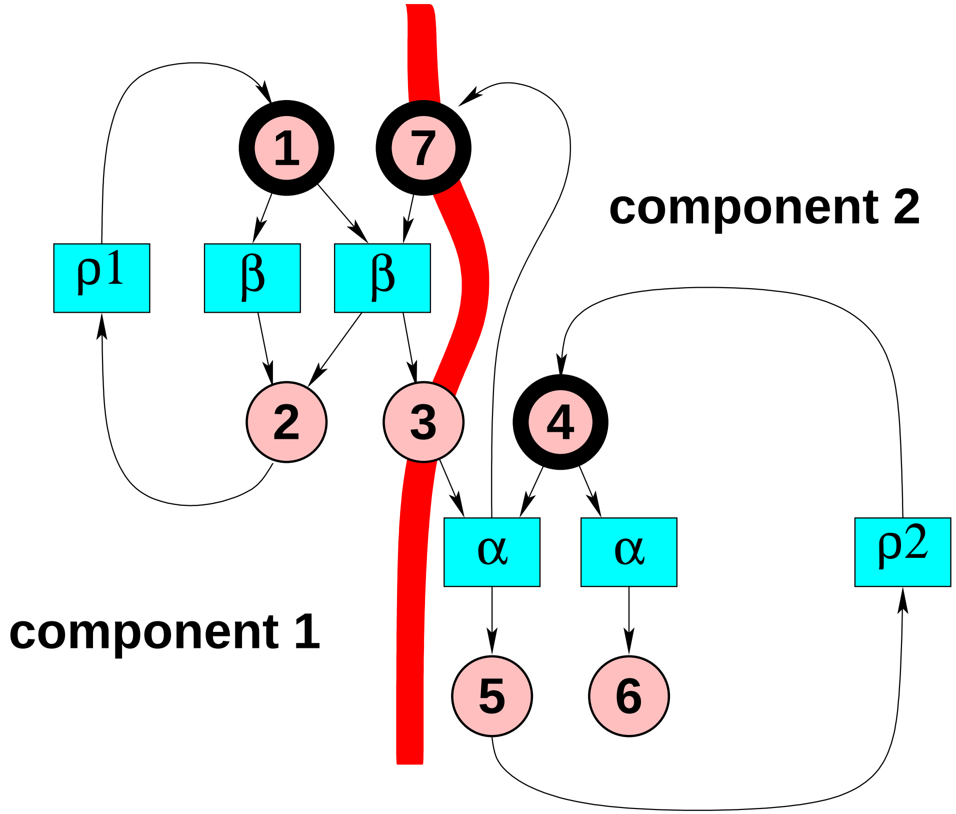


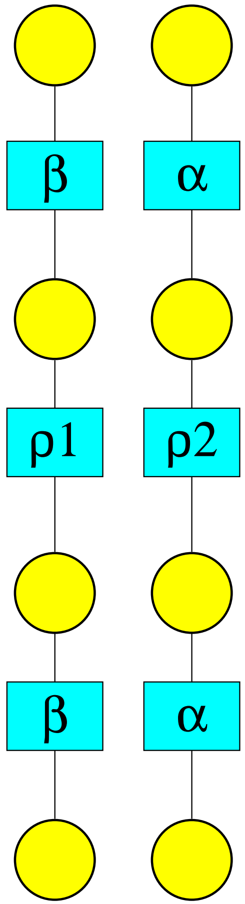
alarm net



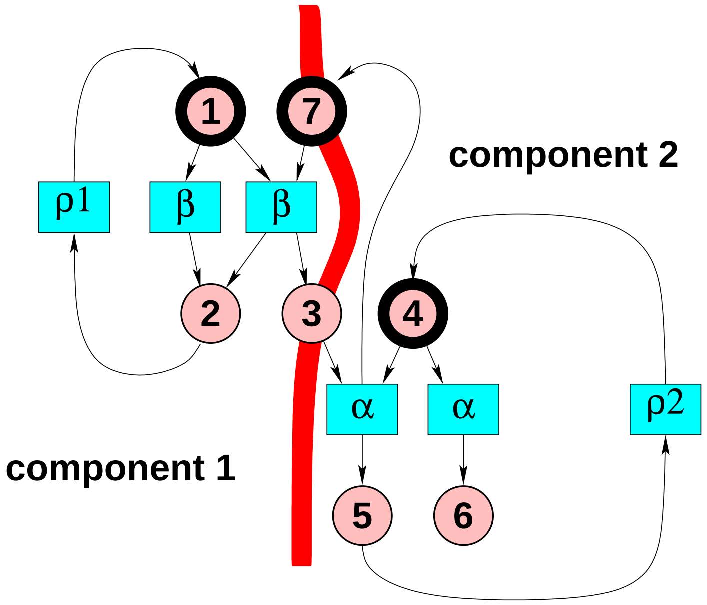
diagnet

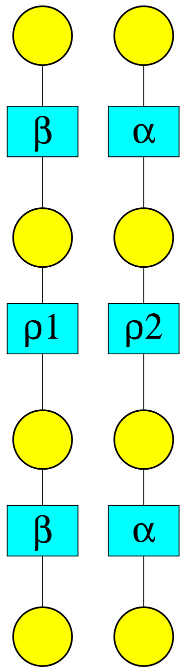
2 interacting components, 2 independent sensors

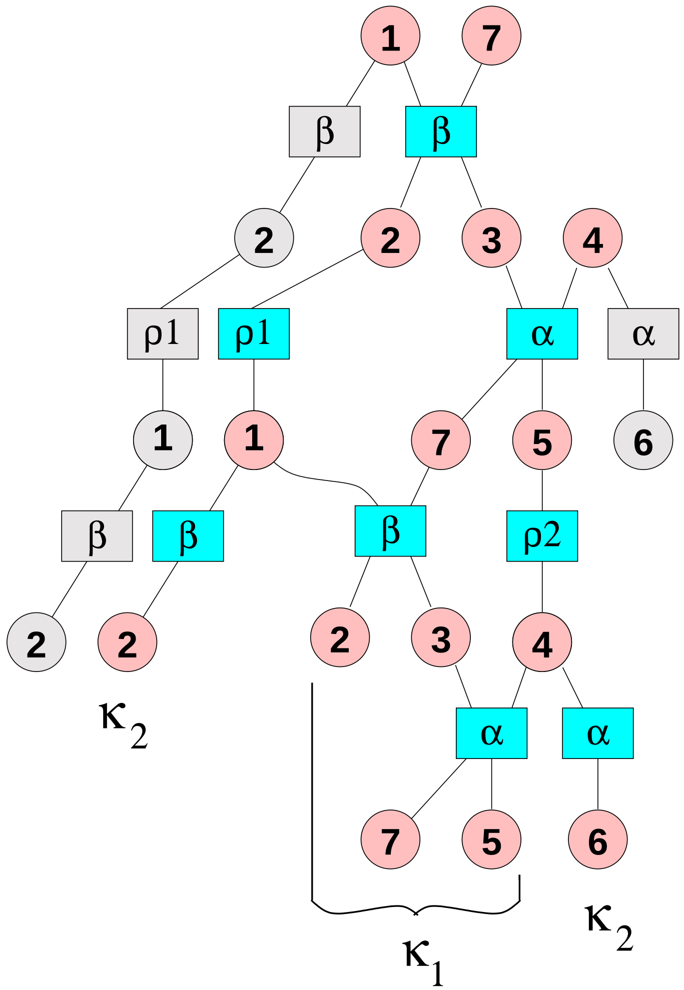




2 components, 2 sensors, 1 supervisor: $\pi_P(\mathcal{U}_{\mathcal{P}\times\mathcal{A}})$







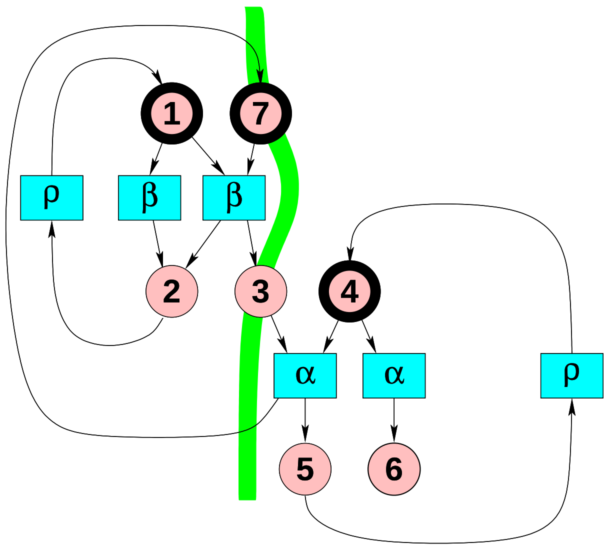
1. A toy example:
    - Petri nets and unfoldings
    - asynchronous diagnosis
    - distributed diagnosis

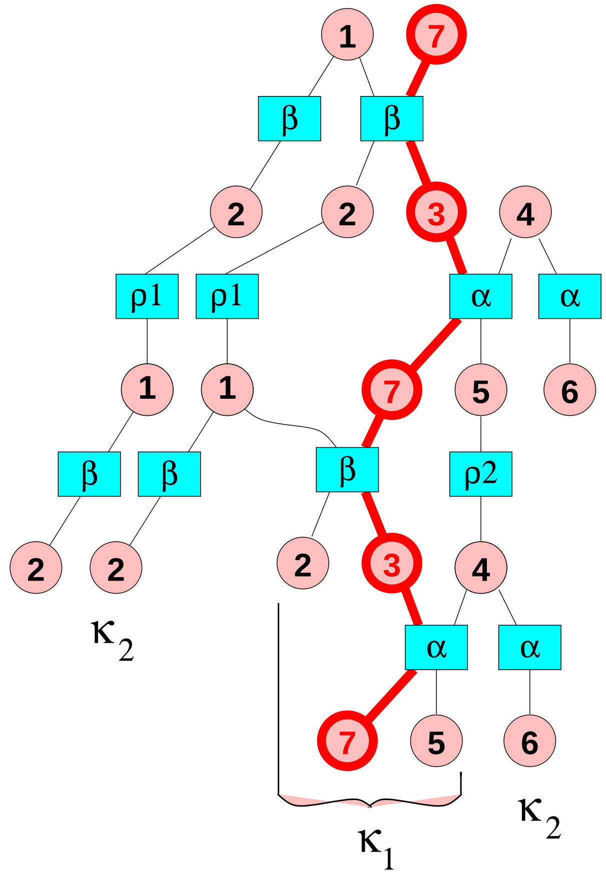
2. Formalizing: Petri nets, unfoldings and event structures

3. An abstract setting

4. Distributed orchestration:
    - tree-shaped networks
    - general networks

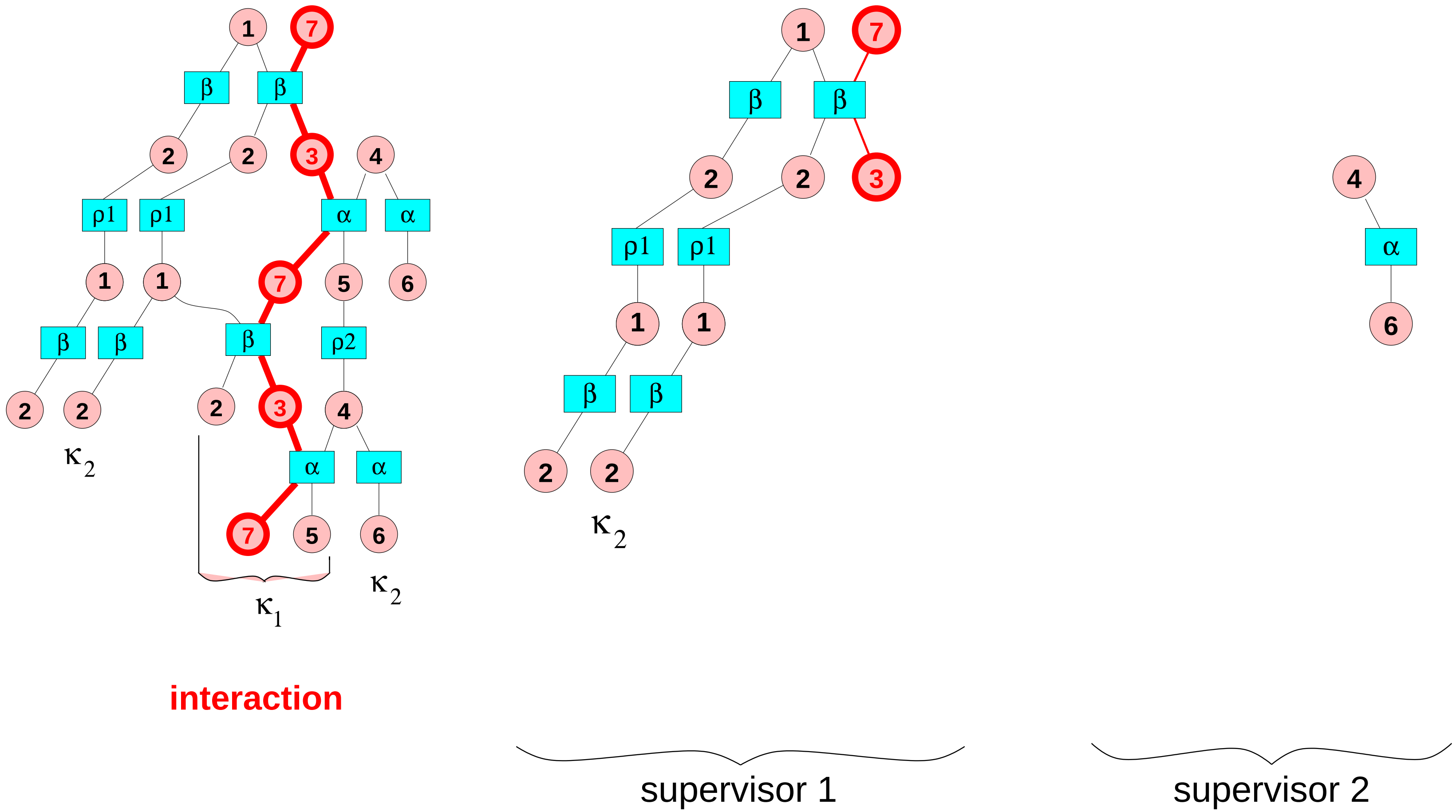
2 components, 2 sensors, 2 supervisors



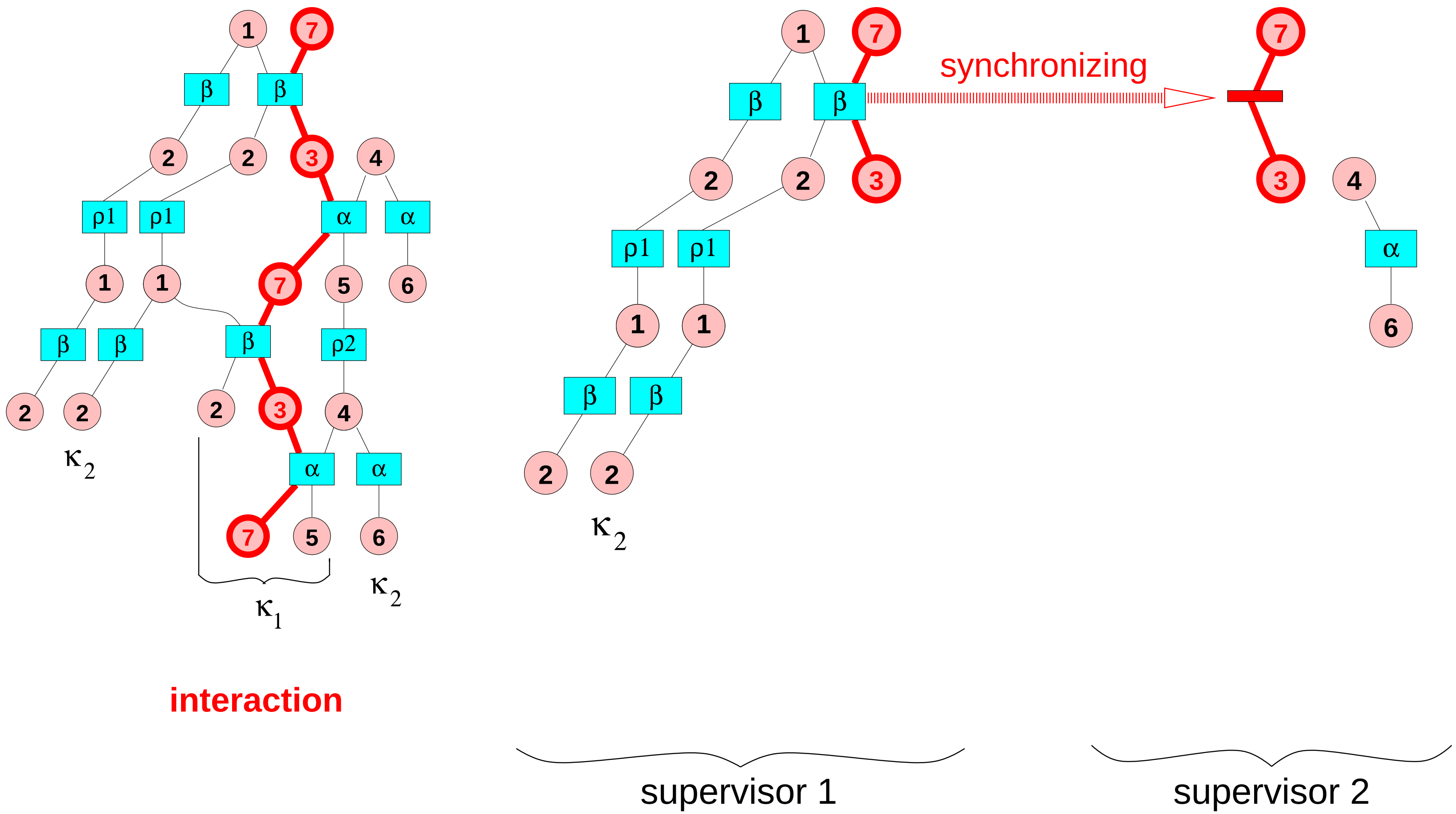


interaction

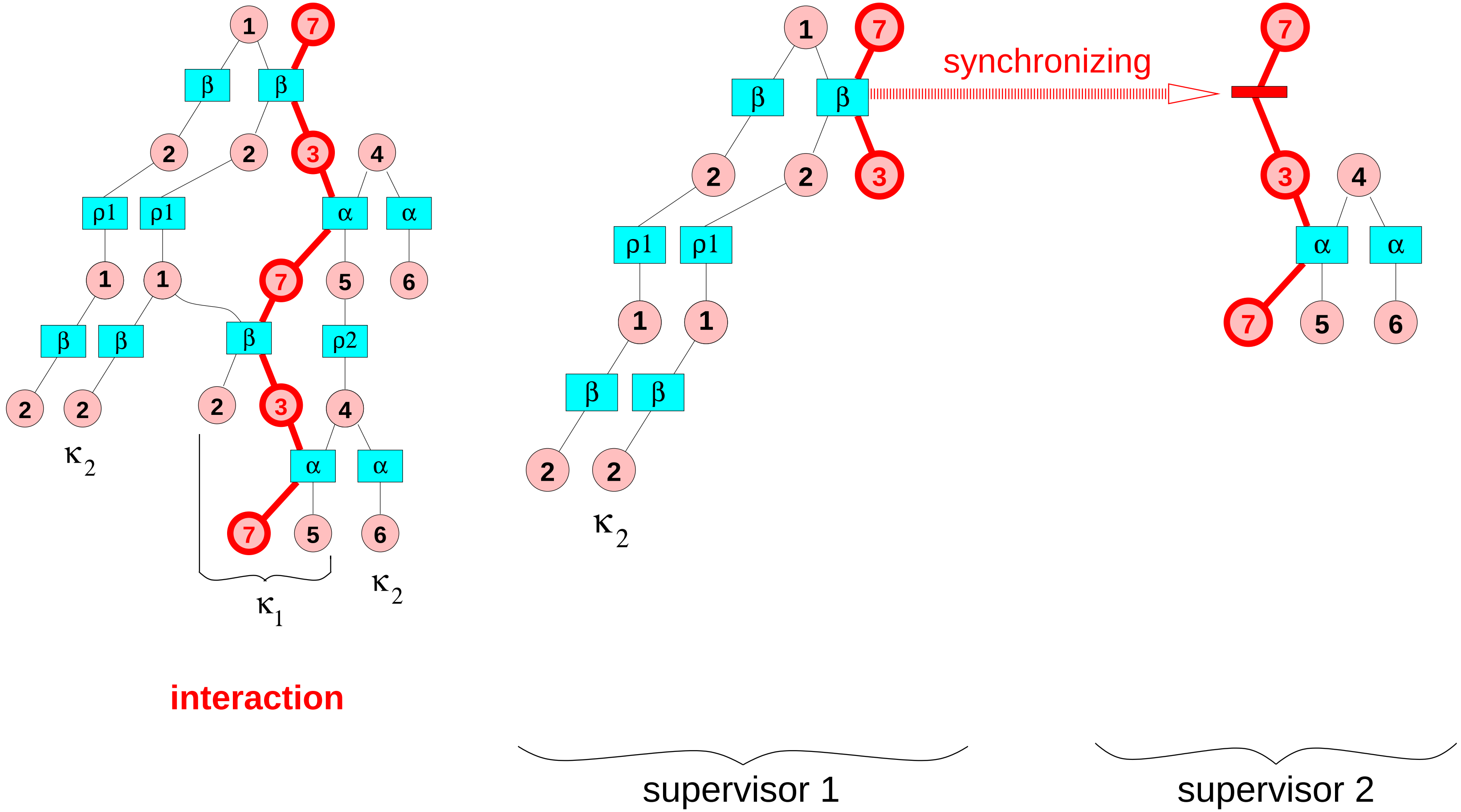
2 components, 2 sensors, 2 supervisors



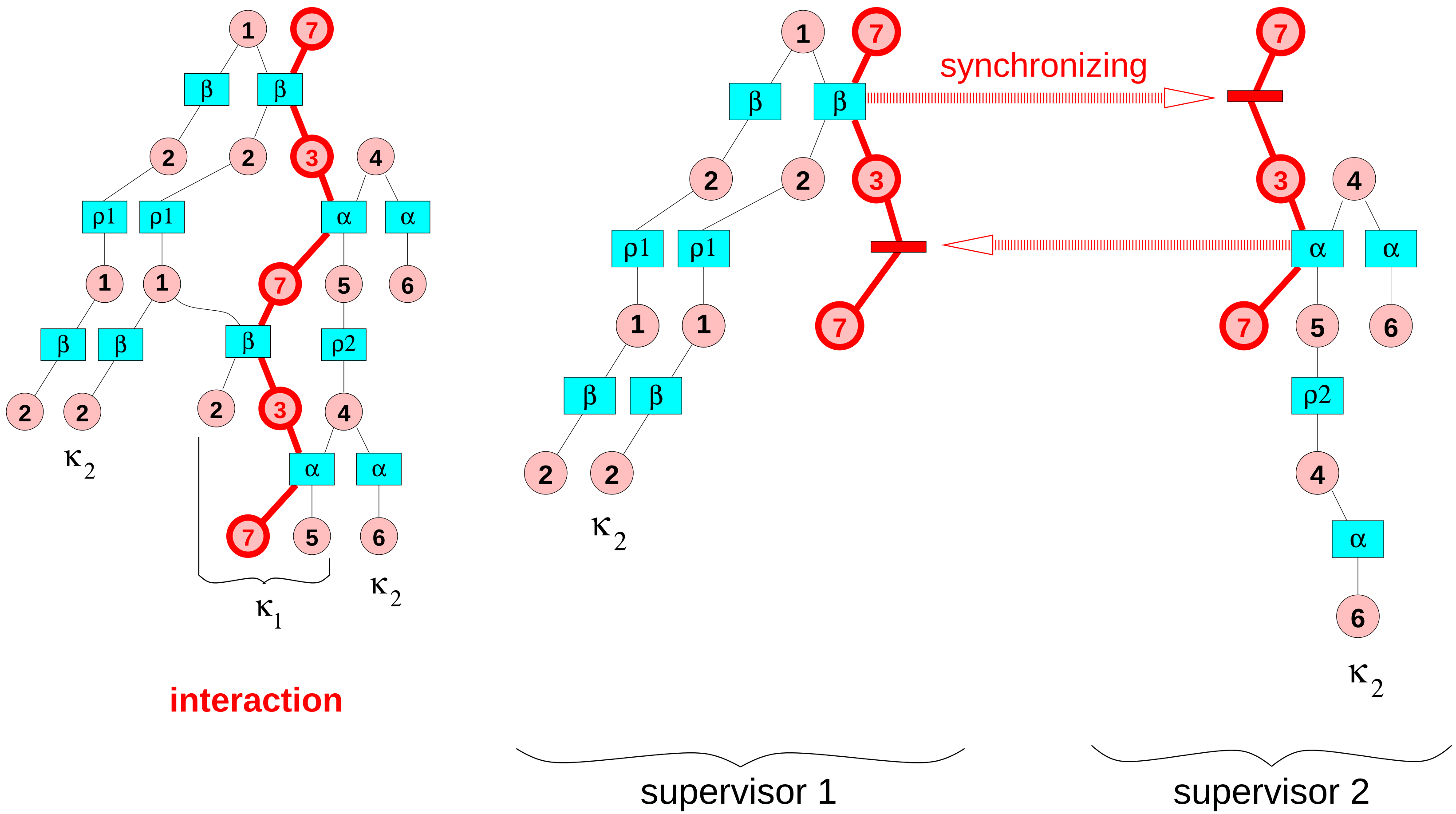
2 **components**, 2 **sensors**, *distributed* **diagnosis**



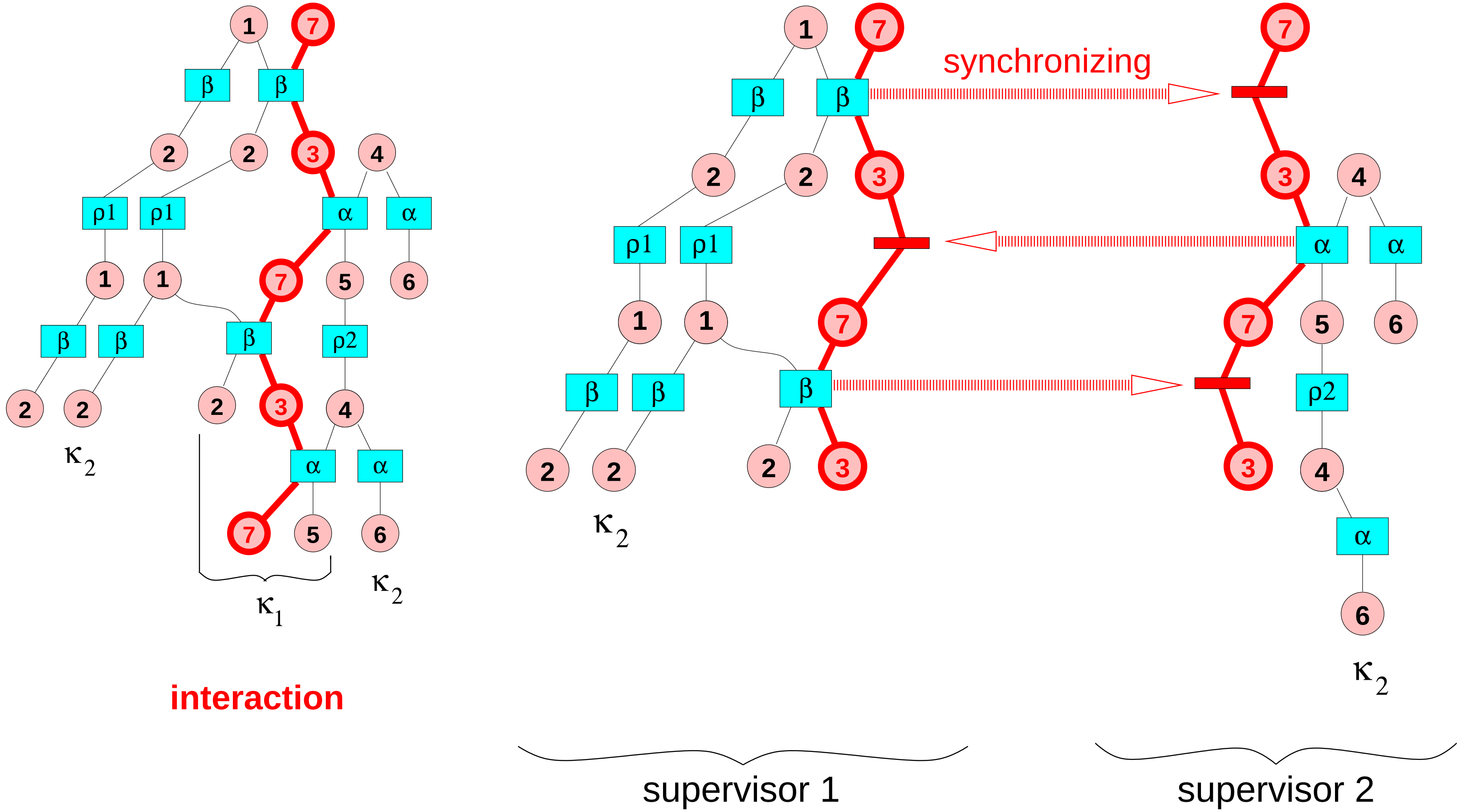
2 **components**, 2 **sensors**, *distributed* **diagnosis**



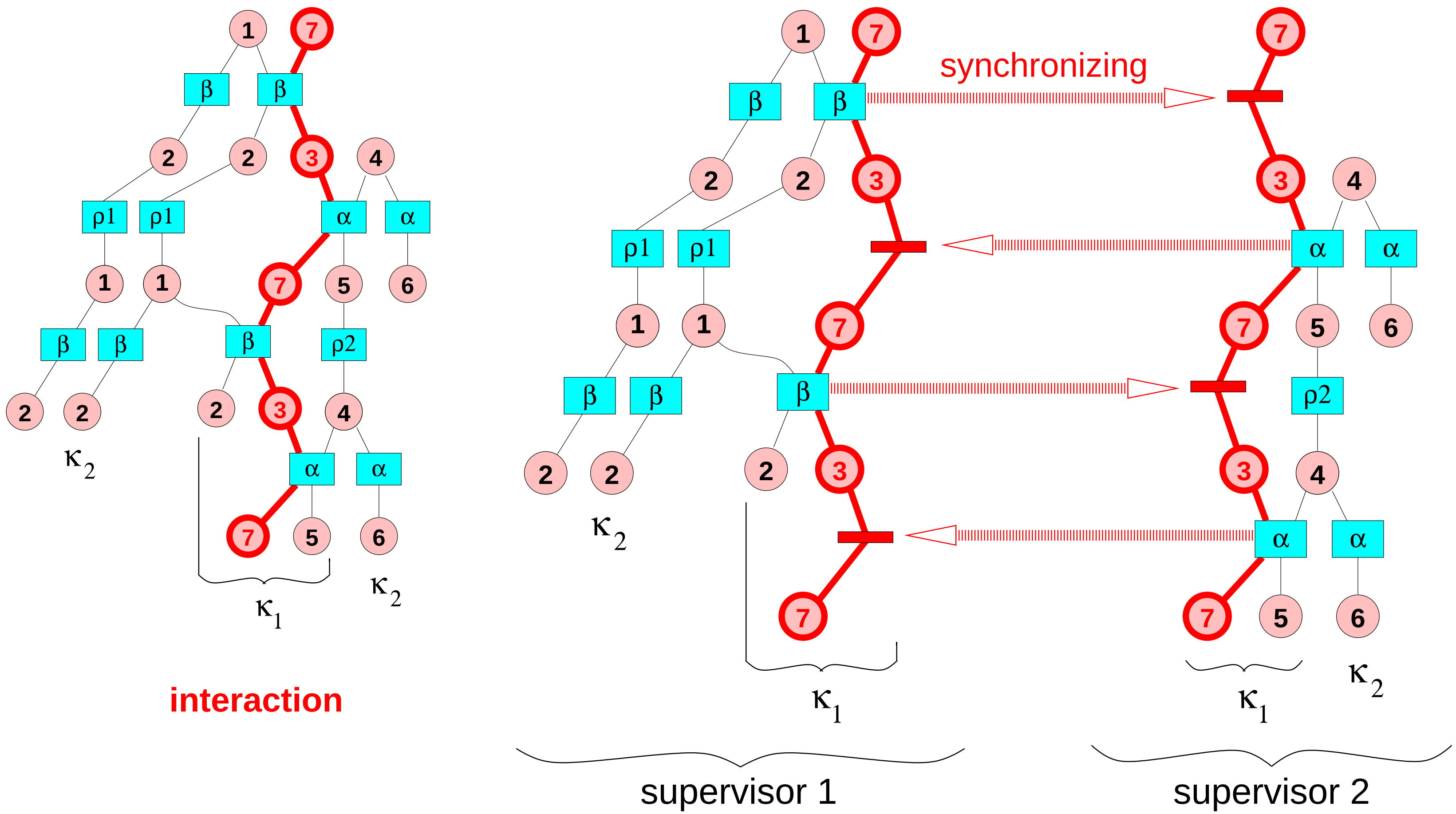
2 **components,** 2 **sensors,** *distributed* **diagnosis**



2 **components**, 2 **sensors**, *distributed* **diagnosis**



2 **components**, 2 **sensors**, *distributed* **diagnosis**



Discussion

local diagnosis is never blocked
each supervisor emits and forgets: write is non-blocking
asynchronous distributed algorithm: no synchronization service

Discussion

local diagnosis is never blocked
each supervisor emits and forgets: write is non-blocking
asynchronous distributed algorithm: no synchronization service

more than 2 supervisors
more complex interaction } ⇒ **very complex algorithm!**

needed :
- formalizing synchronizations & projections of unfoldings
- formalizing the high-level "orchestration"

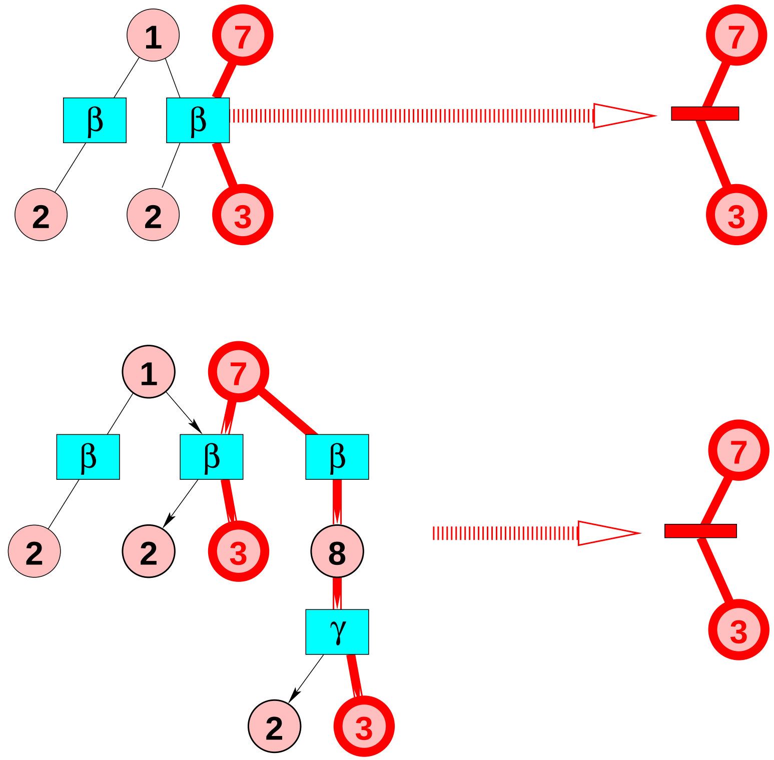
1. A toy example:
   - Petri nets and unfoldings
   - asynchronous diagnosis
   - distributed diagnosis

2. Formalizing: Petri nets, unfoldings and event structures

3. An abstract setting

4. Distributed orchestration:
   - tree-shaped networks
   - general networks

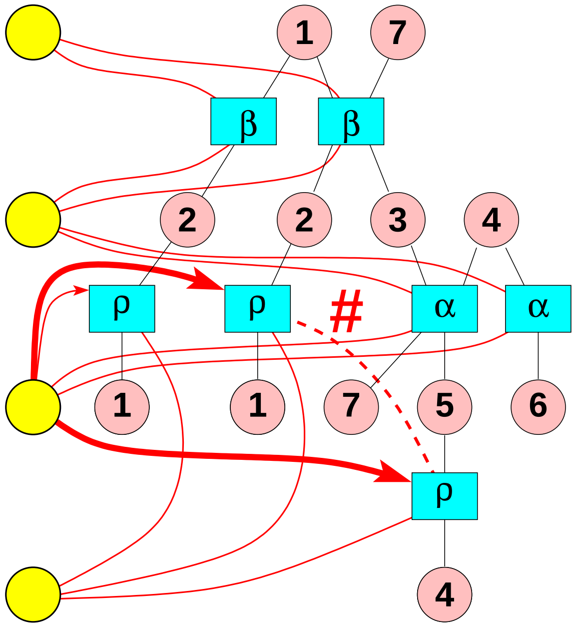
abstractions/projections perform compression



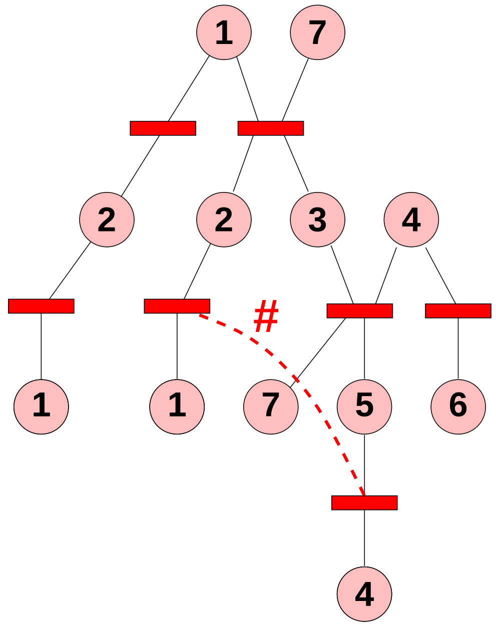
$$\mathcal{U}_{\mathcal{P}} \longmapsto \pi_{P_2}(\mathcal{U}_{\mathcal{P}}) = \{E, \preceq, \#, \varphi\}$$

unfolding *event structure*

abstractions/projections perform compression



diagnet

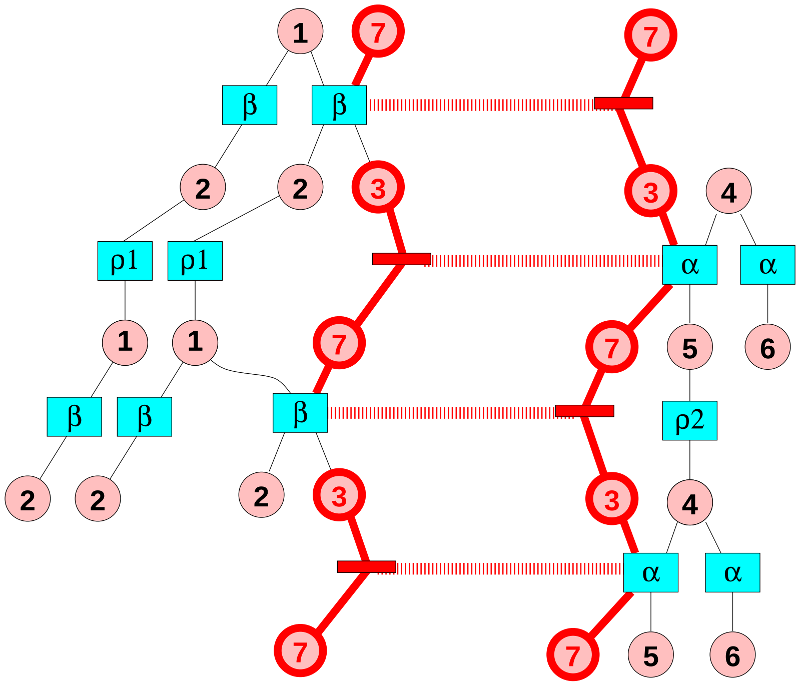


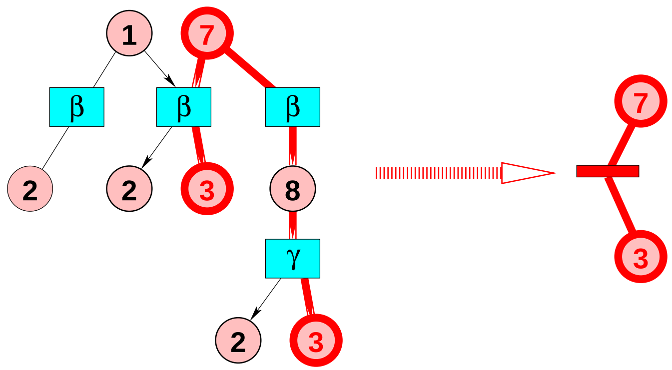
diagnet as an
event structure

$$\mathcal{U}_{\mathcal{P}} \quad \longmapsto \quad \pi_{P_2}(\mathcal{U}_{\mathcal{P}}) = \{E, \preceq, \#, \varphi\}$$

unfolding *event structure*

synchronization $\mathcal{U}_1 \wedge \mathcal{U}_2$; projection $\pi_Q(\mathcal{U})$





regard $\mathcal{U}$'s as event structures

1. A toy example:
   - Petri nets and unfoldings
   - asynchronous diagnosis
   - distributed diagnosis

2. Formalizing: Petri nets, unfoldings and event structures

3. **An abstract setting**

4. Distributed orchestration:
   - tree-shaped networks
   - general networks

synchronization $\mathcal{U}_1 \wedge \mathcal{U}_2$; projection $\pi_Q(\mathcal{U}_\mathcal{P})$

$$P, Q : \ \ label \ \ sets$$

$$\pi \ \ is \ \ a \ \ projection : \pi_P \circ \pi_Q \ = \ \pi_{P \cap Q}$$

$$Q \supseteq P_1 \cap P_2 \ : \ \pi_Q(\mathcal{U}_1 \wedge \mathcal{U}_2) \ = \ \pi_Q(\mathcal{U}_1) \wedge \pi_Q(\mathcal{U}_2)$$

(similar to constraints)

synchronization $\mathcal{U}_1 \wedge \mathcal{U}_2$; projection $\pi_Q\,(\mathcal{U}_{\mathcal{P}})$

$$P, Q : \;\; label \;\; sets$$

$$\pi \;\; is \;\; a \;\; projection : \pi_P \circ \pi_Q \;=\; \pi_{P\cap Q}$$

$$Q \supseteq P_1 \cap P_2 \;\; : \;\; \pi_Q\,(\mathcal{U}_1 \wedge \mathcal{U}_2) \;=\; \pi_Q\,(\mathcal{U}_1) \wedge \pi_Q\,(\mathcal{U}_2)$$

(similar to constraints)

$$\underbrace{\pi_{P_1}\left(\mathcal{U}_{\mathcal{P}_1\|\mathcal{P}_2}\right)}_{local \;\; view} \bigwedge \underbrace{\pi_{P_2}\left(\mathcal{U}_{\mathcal{P}_1\|\mathcal{P}_2}\right)}_{local \;\; view} \;=\; \mathcal{U}_{\mathcal{P}_1\|\mathcal{P}_2}$$

$$please \;\; note : \mathcal{U}_{\mathcal{P}_1} \wedge \mathcal{U}_{\mathcal{P}_2} \;\neq\; \mathcal{U}_{\mathcal{P}_1\|\mathcal{P}_2}$$

synchronization $\mathcal{U}_1 \wedge \mathcal{U}_2$; projection $\pi_Q(\mathcal{U}_\mathcal{P})$

$$P, Q : \ \textit{label sets}$$

$$\pi \ \textit{is a projection} : \pi_P \circ \pi_Q = \pi_{P \cap Q}$$

$$Q \supseteq P_1 \cap P_2 \ : \ \pi_Q(\mathcal{U}_1 \wedge \mathcal{U}_2) = \pi_Q(\mathcal{U}_1) \wedge \pi_Q(\mathcal{U}_2)$$

(similar to constraints)

$$\underbrace{\pi_{P_1}\left(\mathcal{U}_{\mathcal{P}_1 \| \mathcal{P}_2}\right)}_{\textit{local view}} \bigwedge \underbrace{\pi_{P_2}\left(\mathcal{U}_{\mathcal{P}_1 \| \mathcal{P}_2}\right)}_{\textit{local view}} = \mathcal{U}_{\mathcal{P}_1 \| \mathcal{P}_2}$$

$$\textit{please note} : \mathcal{U}_{\mathcal{P}_1} \wedge \mathcal{U}_{\mathcal{P}_2} \neq \mathcal{U}_{\mathcal{P}_1 \| \mathcal{P}_2}$$

distributed diagnosis : $\left[\pi_{P_i}(\mathcal{U}_{\mathcal{P} \times \mathcal{A}})\right]_{i=1,2}$

1. A toy example:
    - Petri nets and unfoldings
    - asynchronous diagnosis
    - distributed diagnosis

2. Formalizing: Petri nets, unfoldings and event structures

3. An abstract setting

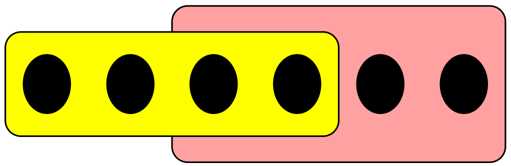
4. **Distributed orchestration:**
    - **tree-shaped networks**
    - general networks

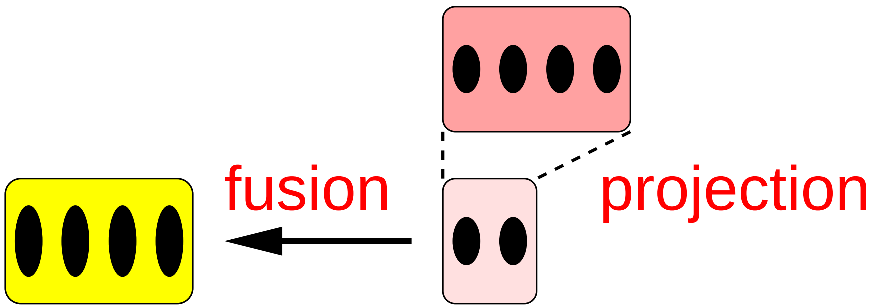
A simple constraint problem

compute $\pi_{P_1}(\mathcal{U}_1 \wedge \mathcal{U}_2)$ **without computing** $\mathcal{U}_1 \wedge \mathcal{U}_2$

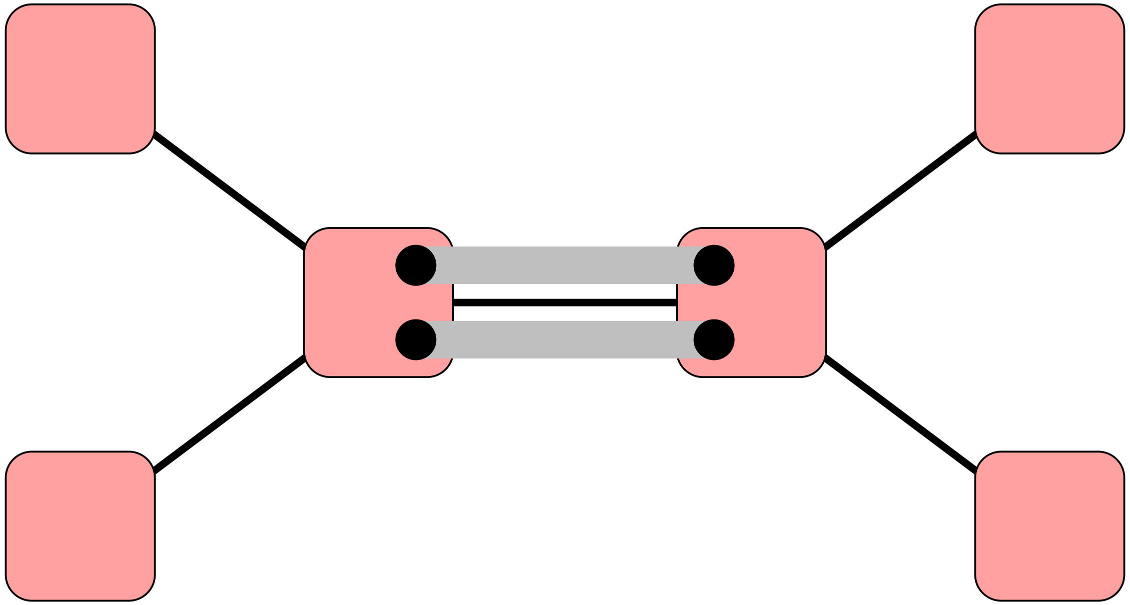
$$\begin{aligned} \pi_P \circ \pi_Q &= \pi_{P \cap Q} \\ Q \supseteq P_1 \cap P_2 \ : \ \pi_Q(\mathcal{U}_1 \wedge \mathcal{U}_2) &= \pi_Q(\mathcal{U}_1) \wedge \pi_Q(\mathcal{U}_2) \end{aligned}$$

$$\begin{aligned} \pi_{P_1}(\mathcal{U}_1 \wedge \mathcal{U}_2) &= \pi_{P_1}(\mathcal{U}_1) \wedge \pi_{P_1}(\mathcal{U}_2) = \mathcal{U}_1 \wedge \pi_{P_1}(\mathcal{U}_2) \\ &= \mathcal{U}_1 \wedge \pi_{P_1} \circ \pi_{P_2}(\mathcal{U}_2) = \underbrace{\mathcal{U}_1 \wedge \underbrace{\pi_{P_1 \cap P_2}(\mathcal{U}_2)}_{\text{projection}}}_{\text{fusion}} \end{aligned}$$



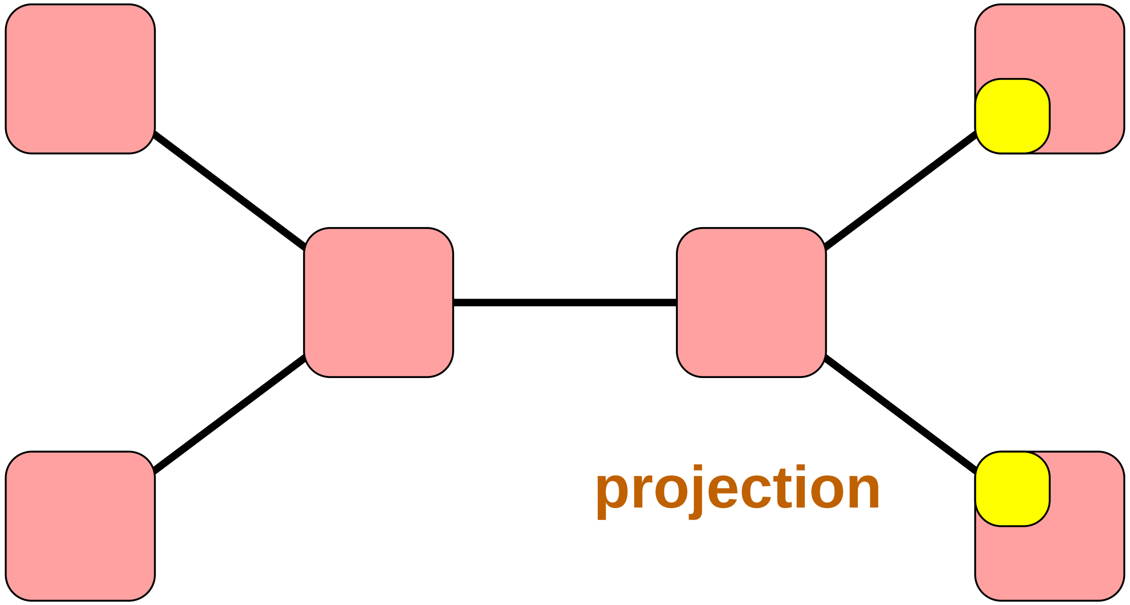


A tree-shaped network of components

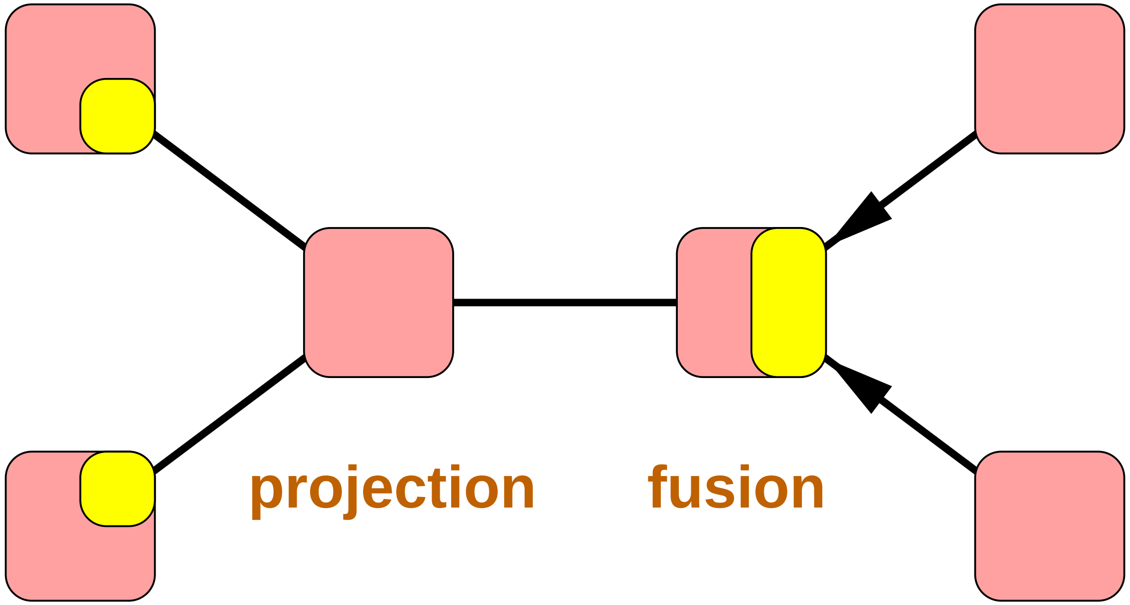


? : $\pi_{P_i}\left(\bigwedge_j \mathcal{U}_j\right)$

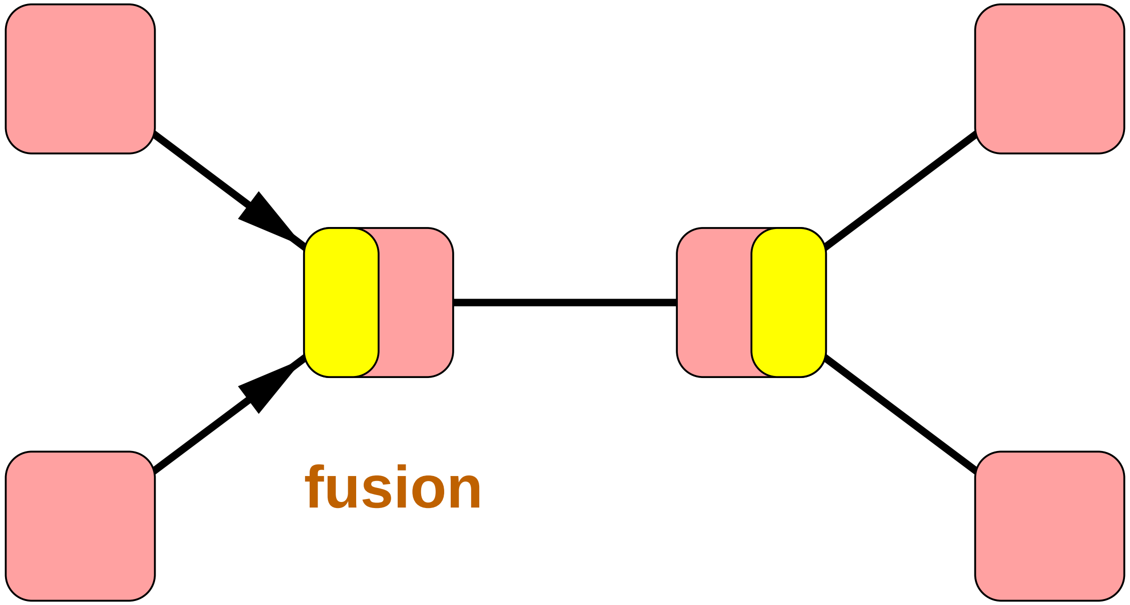
A tree-shaped network of components



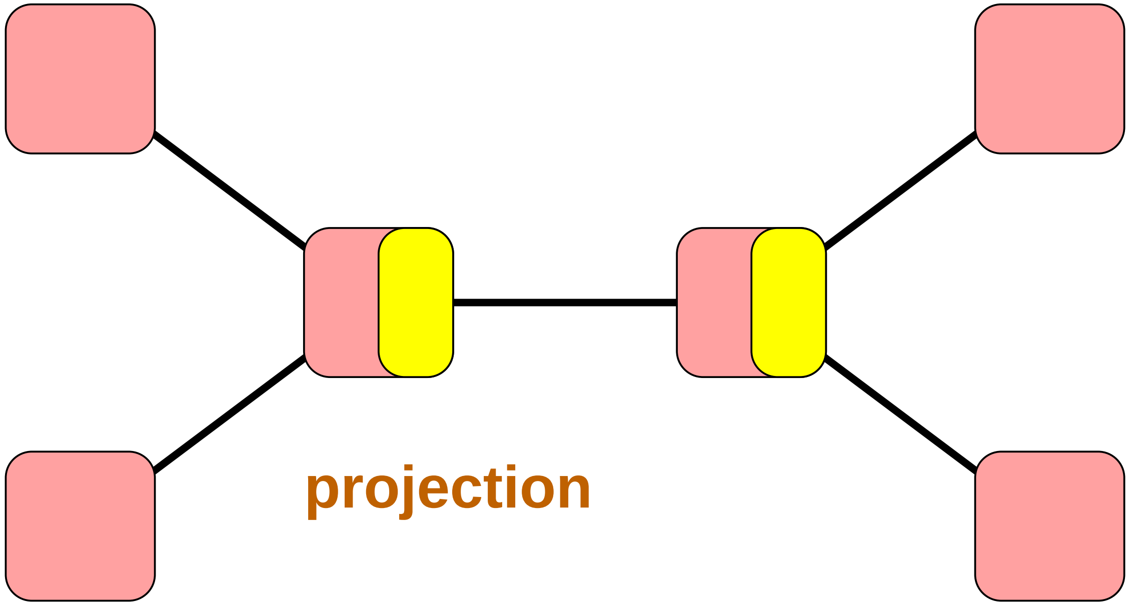
A tree-shaped network of components



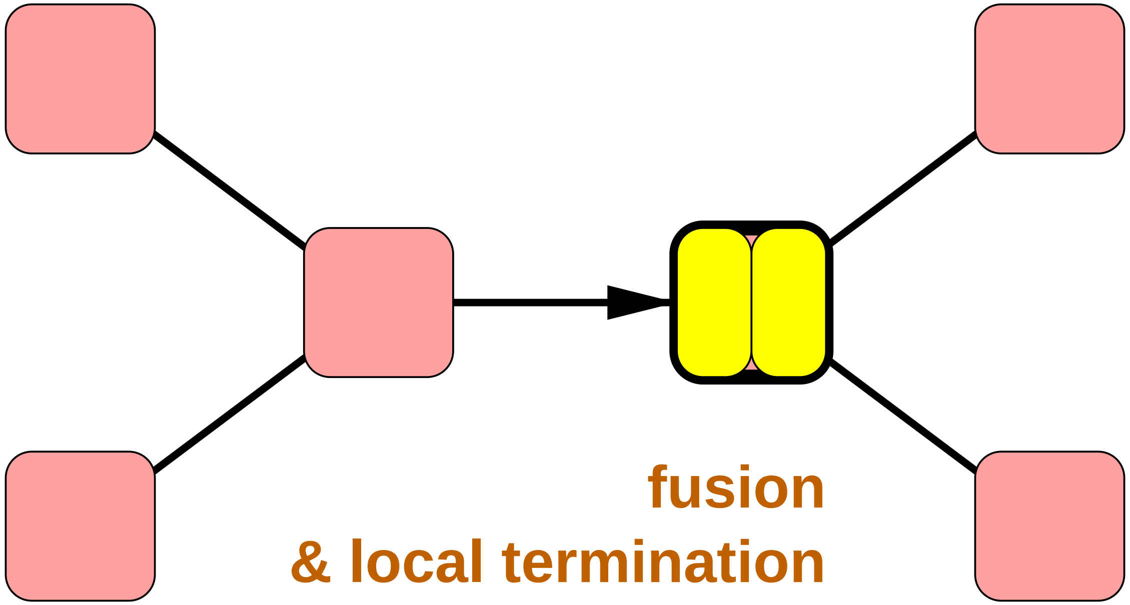
A tree-shaped network of components



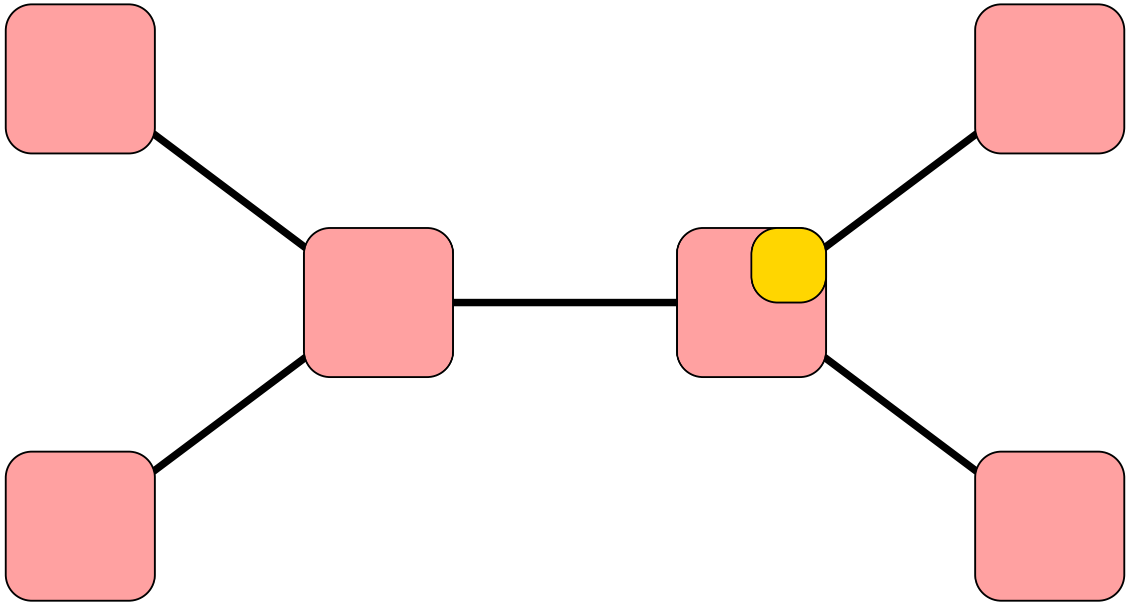
A tree-shaped network of components



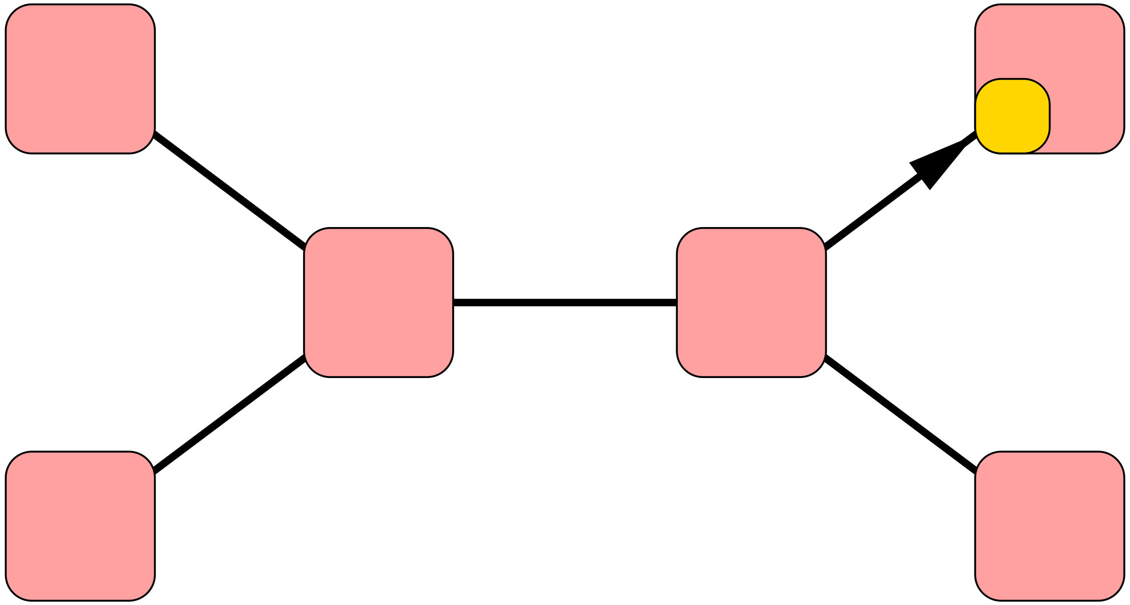
A tree-shaped network of components



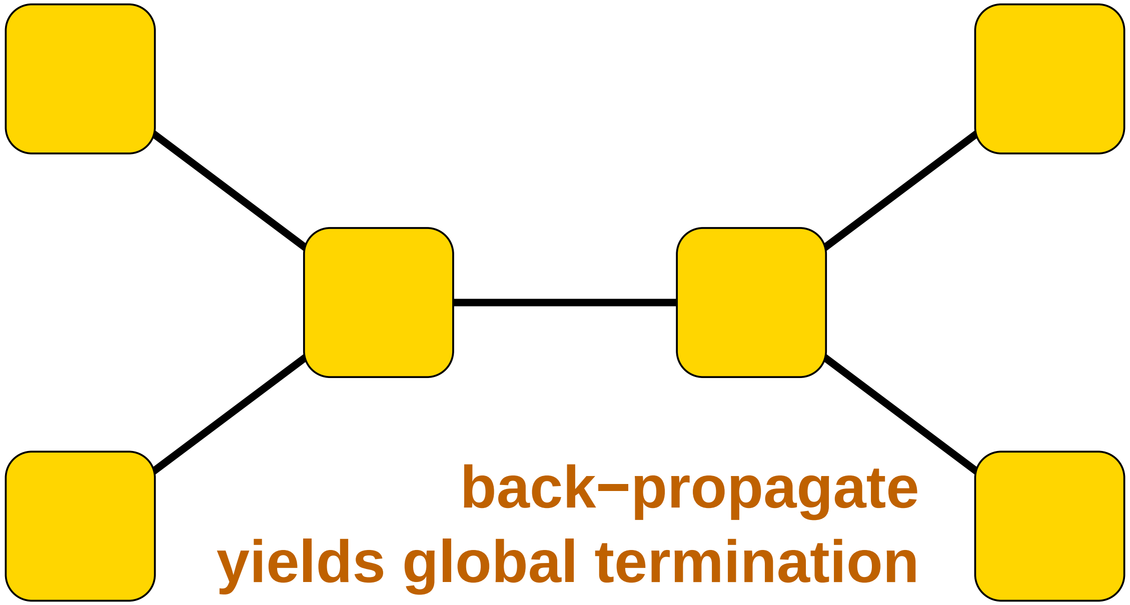
A tree-shaped network of components



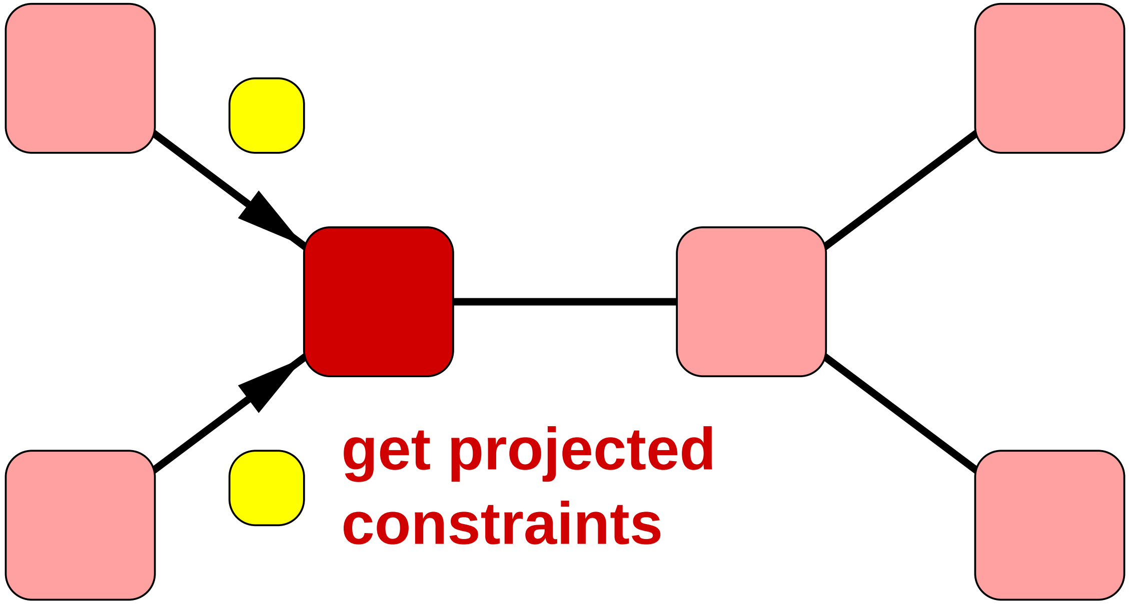
A tree-shaped network of components



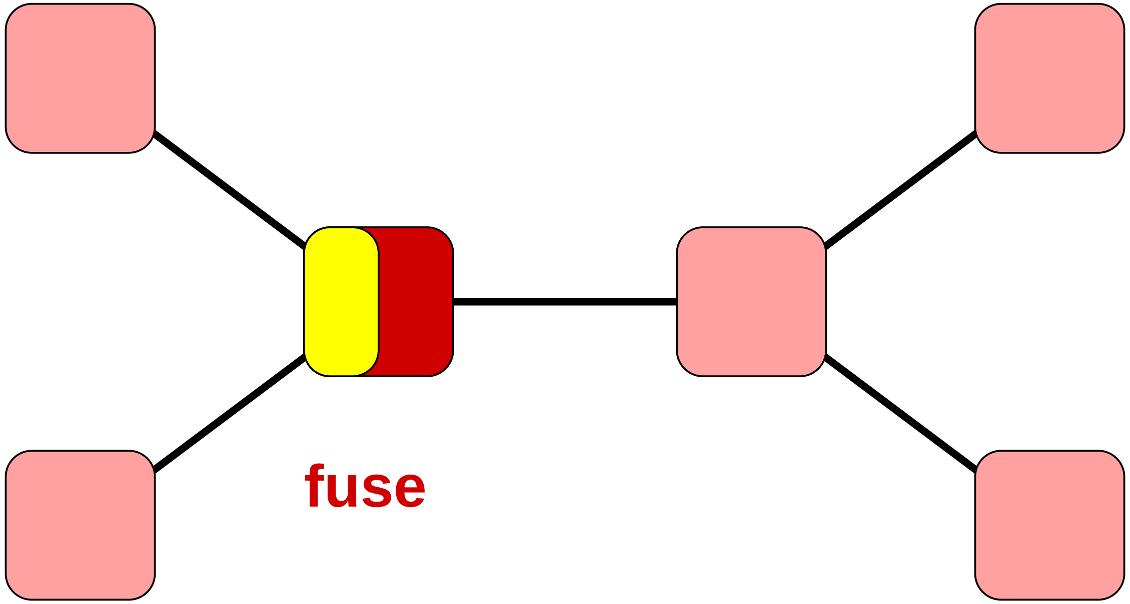
A tree-shaped network of components



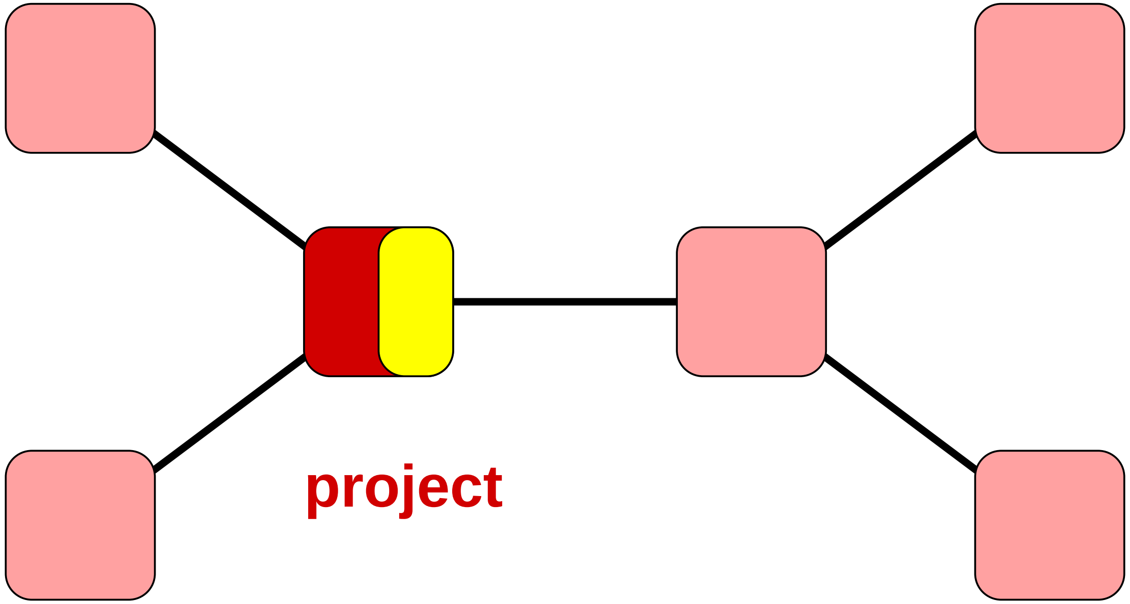
A chaotic algorithm



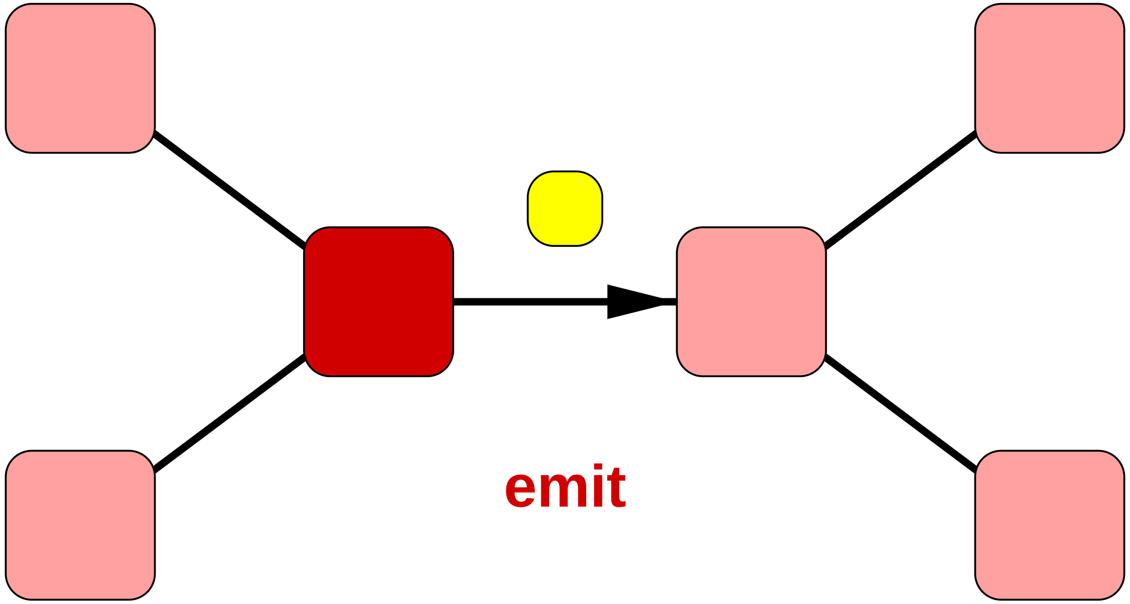
A chaotic algorithm



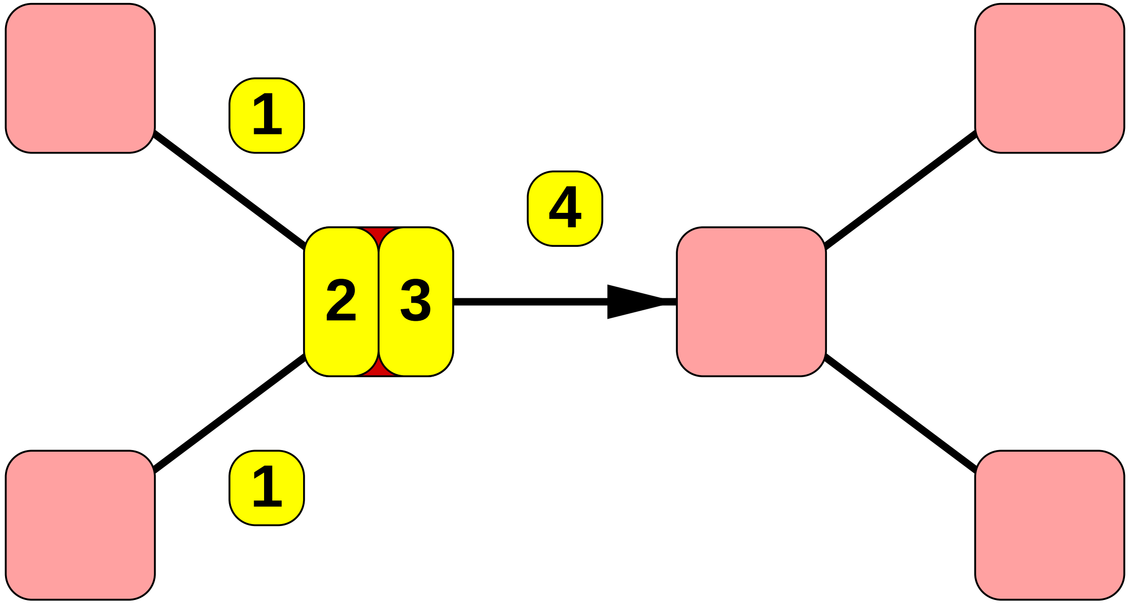
A chaotic algorithm



A chaotic algorithm



A chaotic algorithm



each node performs this atomic sequence of micro-steps concurrently, in a chaotic way; messages travel along the branches

A theorem for tree-shaped networks

The initial conditions are the $\mathcal{U}_i$.

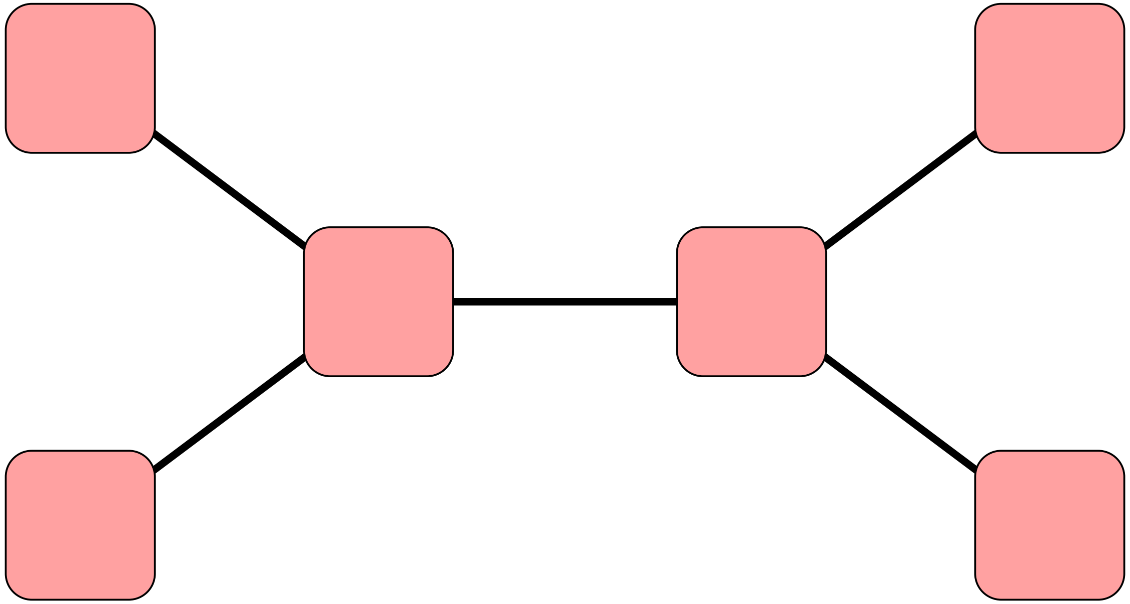
A theorem for tree-shaped networks

The initial conditions are the $\mathcal{U}_i$. The iterations apply in a chaotic way. Termination occurs when all messages become stationary.

A theorem for tree-shaped networks

The initial conditions are the $\mathcal{U}_i$. The iterations apply in a chaotic way. Termination occurs when all messages become stationary. Yields the desired solution $\pi_{P_i}\left(\bigwedge_j \mathcal{U}_j\right)$

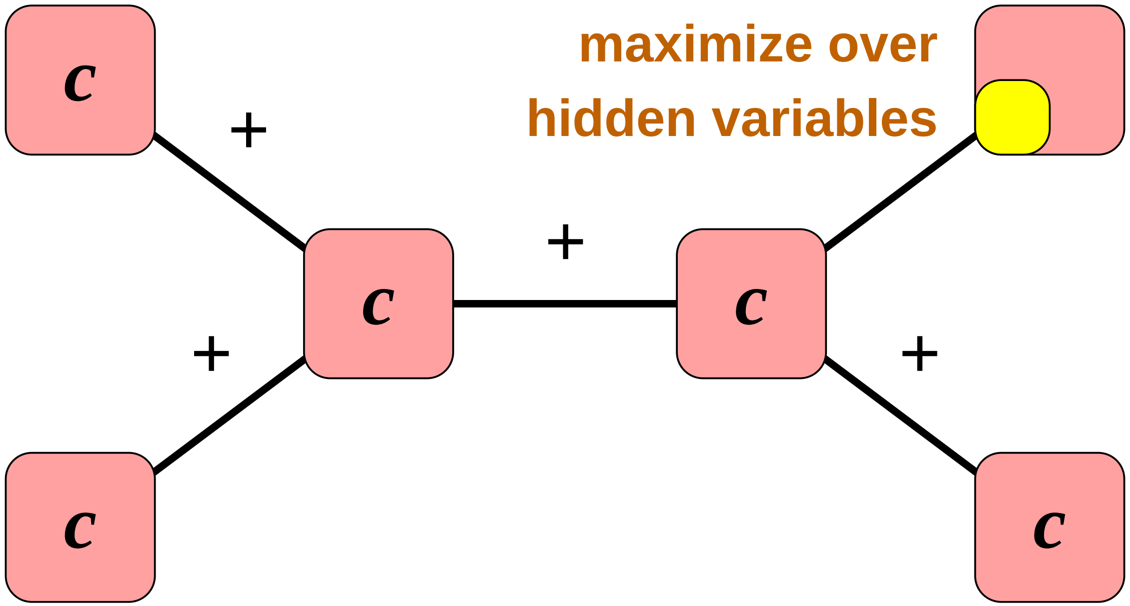
Extension to time-varying systems



$$\textbf{?} : \pi_{P_i}\left(\bigwedge_j \mathcal{U}_j(n)\right), \mathcal{U}_j(n) \searrow$$

works, thanks to monotonicity of the algorithm!
works even on-line, if messages are fast enough.
Solves on-line diagnosis.

Extension to optimization & belief nets



Solutions can be given an additive cost for minimization (axioms still valid). Can be interpreted as a likelihood for belief nets: belief propagation. Extends also the two-point boundary smoothing algorithms from control.

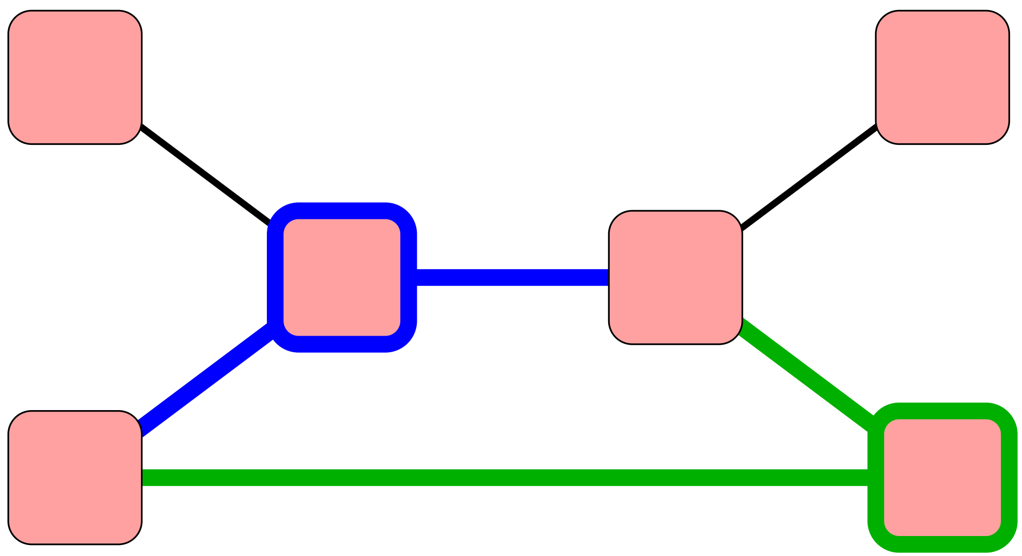
1. A toy example:
   - Petri nets and unfoldings
   - asynchronous diagnosis
   - distributed diagnosis

2. Formalizing: Petri nets, unfoldings and event structures

3. An abstract setting

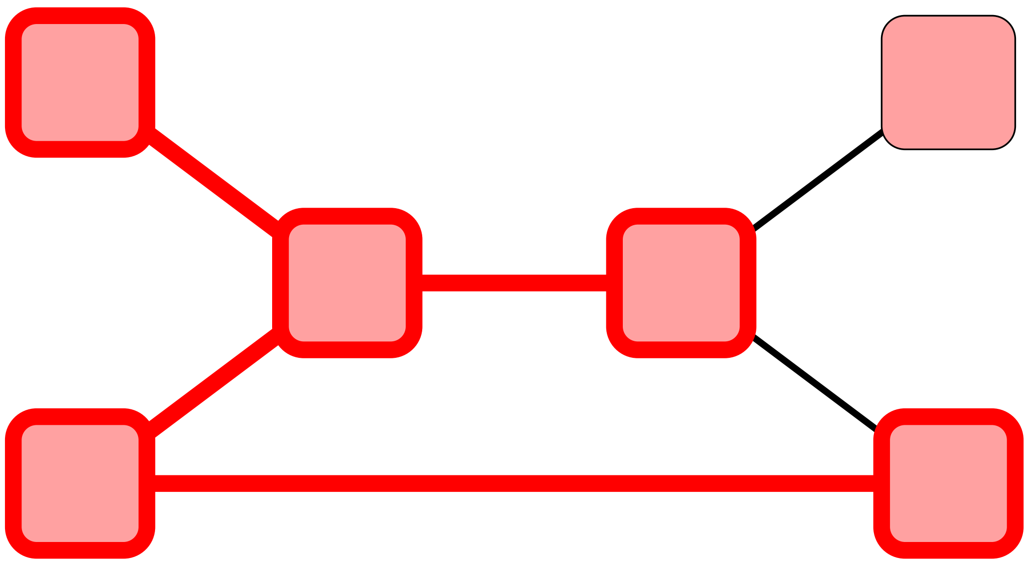
4. Distributed orchestration:
   - tree-shaped networks
   - general networks

Networks with cycles



problem: interaction between distant nodes through different paths ⇒ causality & conflict travel through different paths ⇒ chaotic algorithm invalid in general

Networks with cycles



problem: interaction between distant nodes through different paths $\Rightarrow$ causality & conflict travel through different paths $\Rightarrow$ chaotic algorithm invalid in general

still, this algorithm finds all solutions having tree-shaped support

CONCLUSION

- Computing a local view of global diagnosis without computing global diagnosis
- Expressed using unfoldings $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$, their composition $\bigwedge$, and their projections π_P
- Abstract setting: distributed constraint solving
- Orchestration as a chaotic, distributed iteration

- A prototype developed using Java threads was subsequently deployed as such on a distributed management platform at Alcatel

CONCLUSION

- Computing a local view of global diagnosis without computing global diagnosis
- Expressed using unfoldings $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$, their composition $\bigwedge$, and their projections π_P
- Abstract setting: distributed constraint solving
- Orchestration as a chaotic, distributed iteration

- A prototype developed using Java threads was subsequently deployed as such on a distributed management platform at Alcatel
- Generalizes: optimization, negociation
- Further issues: (graph grammars & unfoldings)
  dynamic reconfiguration
  self-management, Web services

RELATED TOPICS

- network & service management
- distributed algorithms
- fault tolerance
- Discrete Event Systems control and diagnosis
- Hidden Markov Models (HMM), Belief nets, Markov random fields in probability and AI
- Turbo coding in information theory