Distributed diagnosis of concurrent and asynchronous Discrete Event Systems

Albert Benveniste
Eric Fabre, Stefan Haar, Claude Jard
IRISA, Rennes

2004
The problem:

collecting alarms

exchanging messages

computing a local view of global diagnosis
The problem: distributed + asynchronous

- collecting alarms
- exchanging messages
- computing a local view of global diagnosis
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
- 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
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 structure to represent sets of traces with concurrency
a structure to represent sets of traces with concurrency
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: $\mathcal{P}, \mathcal{A}$
Diagnets: $\mathcal{P}, \mathcal{A}, \mathcal{U}_{\mathcal{P} \times \mathcal{A}}$
Diagnets: \( P, A, \pi_P(U_P \times A) \)
2 interacting components, 2 independent sensors
2 components, 2 sensors, 1 supervisor: $\pi_P(\mathcal{U}_{P \times A})$
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
2 components, 2 sensors, 2 supervisors
2 components, 2 sensors, distributed diagnosis

interaction

supervisor 1

supervisor 2
2 components, 2 sensors, distributed diagnosis

interaction

supervisor 1

supervisor 2
2 components, 2 sensors, distributed diagnosis

interaction

synchronizing

supervisor 1

supervisor 2
2 components, 2 sensors, distributed diagnosis

interaction

supervisor 1

supervisor 2
2 components, 2 sensors, distributed diagnosis

interaction

supervisor 1

supervisor 2
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 \[ \Rightarrow \] very complex algorithm!

needed: \[
\begin{align*}
\text{formalizing synchronizations \& projections} \\
\text{of unfoldings} \\
\text{formalizing the high-level “orchestration”}
\end{align*}
\]
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

\[ U_P \mapsto \pi_{P_2}(U_P) = \{ E, \preceq, \# , \varphi \} \]

unfolding \hspace{1cm} event structure
abstractions/projections perform compression

\[ U_P \overset{\text{unfolding}}{\longrightarrow} \pi_{P_2}(U_P) = \{ E, \leq, \#, \varphi \} \]

event structure
synchronization $U_1 \wedge U_2$; projection $\pi_Q(U)$

regard $U$’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 $U_1 \land U_2$; projection $\pi_Q(U_P)$

$P, Q$: label sets

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

$Q \supseteq P_1 \cap P_2$: $\pi_Q(U_1 \land U_2) = \pi_Q(U_1) \land \pi_Q(U_2)$

(similar to constraints)
synchronization $\mathcal{U}_1 \land \mathcal{U}_2$; projection $\pi_Q(\mathcal{U}_P)$

$P, Q : label \ sets$

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

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

(similar to constraints)

$\pi_{P_1} \left( \mathcal{U}_{P_1 \| P_2} \right) \land \pi_{P_2} \left( \mathcal{U}_{P_1 \| P_2} \right) = \mathcal{U}_{P_1 \| P_2}$

(local view) (local view)

please note: $\mathcal{U}_{P_1 \land P_2} \neq \mathcal{U}_{P_1 \| P_2}$
synchronization \( U_1 \wedge U_2 \); projection \( \pi_Q(U_P) \)

\[ P, Q : \text{label sets} \]
\[
\pi \text{ is a projection : } \pi_P \circ \pi_Q = \pi_{P \cap Q}
\]
\( Q \supseteq P_1 \cap P_2 : \pi_Q(U_1 \wedge U_2) = \pi_Q(U_1) \wedge \pi_Q(U_2) \)

(similar to constraints)

\[
\pi_{P_1}(U_{P_1 \| P_2}) \wedge \pi_{P_2}(U_{P_1 \| P_2}) = U_{P_1 \| P_2}
\]

(local view local view)

please note : \( U_{P_1} \wedge U_{P_2} \neq U_{P_1 \| P_2} \)

distributed diagnosis : \[ [\pi_{P_i}(U_P \times A)]_{i=1,2} \]
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 $\pi_{P_1}(U_1 \land U_2)$ without computing $U_1 \land U_2$

$\pi_P \circ \pi_Q = \pi_{P \cap Q}$

$Q \supseteq P_1 \cap P_2 : \pi_Q(U_1 \land U_2) = \pi_Q(U_1) \land \pi_Q(U_2)$

$\pi_{P_1}(U_1 \land U_2) = \pi_{P_1}(U_1) \land \pi_{P_1}(U_2) = U_1 \land \pi_{P_1}(U_2)$

$= U_1 \land \pi_{P_1 \circ P_2}(U_2) = U_1 \land \pi_{P_1 \cap P_2}(U_2)$

projection fusion
A tree-shaped network of components

? : $\pi_{P_i} \left( \bigwedge_{j} U_j \right)$
A tree-shaped network of components

projection
A tree-shaped network of components

projection  fusion
A tree-shaped network of components
A tree-shaped network of components

projection
A tree-shaped network of components

fusion & local termination
A tree-shaped network of components
A tree-shaped network of components
A tree-shaped network of components

back-propagate yields global termination
A chaotic algorithm

get projected constraints
A chaotic algorithm

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

A chaotic algorithm project
A chaotic algorithm

emit each node performs this atomic sequence of micro-steps concurrently, in a chaotic way; messages travel along the branches.
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 $U_i$. 
A theorem for tree-shaped networks

The initial conditions are the $U_i$. 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 $U_i$. The iterations apply in a chaotic way. Termination occurs when all messages become stationary. Yields the desired solution $\pi_{P_i}(\bigwedge_j U_j)$
Extension to time-varying systems

? : $\pi P_i \left( \bigwedge_j U_j(n) \right)$, $U_j(n) \downarrow$

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
problem: interaction between distant nodes through different paths ⇒ causality & conflict travel through different paths ⇒ chaotic algorithm invalid in general
problem: interaction between distant nodes through different paths ⇒ causality & conflict travel through different paths ⇒ 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 $U_{p \times A}$, their composition $\wedge$, and their projections $\pi_p$
- 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 $\mathcal{U}_{P \times A}$, their composition $\wedge$, and their projections $\pi_P$
- 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