

Causal memory distributed games

Paul Gastin, Benjamin Lerman, Marc Zeitoun

LIAFA, Université Paris 7, UMR CNRS 7089

`{Paul.Gastin,Benjamin.Lerman,Marc.Zeitoun@}liafa.jussieu.fr`

Games meeting, Bordeaux, sept. 2004

Decidability of a class of distributed games

The result of this talk

Asynchronous distributed games with controlled reachability conditions
are decidable
for symmetric series-parallel systems and causal memory strategies.

Main points

- ▶ Asynchronous processes, test-and-set instructions (read-write is atomic)
- ▶ Controlled reachability conditions: encompass reachability, safety. Not liveness
- ▶ Symmetric series-parallel systems: restriction on dependence between players
- ▶ With local memory: games are usually undecidable

[Madhusudan–Thiagarajan]

[Bernet–Janin–Walukiewicz]

Decidability of a class of distributed games

The result of this talk

Asynchronous distributed games with controlled reachability conditions are decidable
for symmetric series-parallel systems and causal memory strategies.

Main points

- ▶ **Asynchronous processes, test-and-set instructions (read-write is atomic)**
- ▶ Controlled reachability conditions: encompass reachability, safety. Not liveness
- ▶ Symmetric series-parallel systems: restriction on dependence between players
- ▶ With local memory: games are usually undecidable

[Madhusudan–Thiagarajan]

[Bernet–Janin–Walukiewicz]

Decidability of a class of distributed games

The result of this talk

Asynchronous distributed games with **controlled reachability conditions** are decidable
for symmetric series-parallel systems and causal memory strategies.

Main points

- ▶ Asynchronous processes, test-and-set instructions (read-write is atomic)
- ▶ **Controlled reachability conditions: encompass reachability, safety. Not liveness**
- ▶ Symmetric series-parallel systems: restriction on dependence between players
- ▶ With local memory: games are usually undecidable

[Madhusudan–Thiagarajan]

[Bernet–Janin–Walukiewicz]

Decidability of a class of distributed games

The result of this talk

Asynchronous distributed games with controlled reachability conditions
are decidable
for **symmetric series-parallel systems** and causal memory strategies.

Main points

- ▶ Asynchronous processes, test-and-set instructions (read-write is atomic)
- ▶ Controlled reachability conditions: encompass reachability, safety. Not liveness
- ▶ **Symmetric series-parallel systems: restriction on dependence between players**
- ▶ With local memory: games are usually undecidable

[Madhusudan–Thiagarajan]

[Bernet–Janin–Walukiewicz]

Decidability of a class of distributed games

The result of this talk

Asynchronous distributed games with controlled reachability conditions are **decidable** for symmetric series-parallel systems and **causal memory strategies**.

Main points

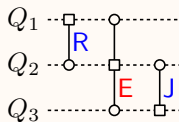
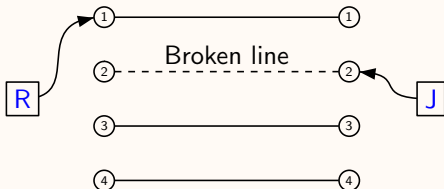
- ▶ Asynchronous processes, test-and-set instructions (read-write is atomic)
- ▶ Controlled reachability conditions: encompass reachability, safety. Not liveness
- ▶ Symmetric series-parallel systems: restriction on dependence between players
- ▶ **With local memory: games are usually undecidable**

[Madhusudan–Thiagarajan] [Bernet–Janin–Walukiewicz]

Example

Romeo and Juliet against the environment

- ▶ Want to communicate through the same communication line
- ▶ At any time, one line is broken
- ▶ Environment looks where R&J are connected, and then, atomically, possibly changes the broken line
- ▶ Romeo/Juliet looks status of lines and, atomically, chooses where to connect.
- ▶ Dependence between moves $R - E - J$
- ▶ If no fairness assumption: E can monopolize the game breaking lines forever



Communication architecture

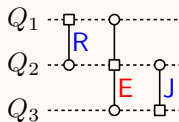
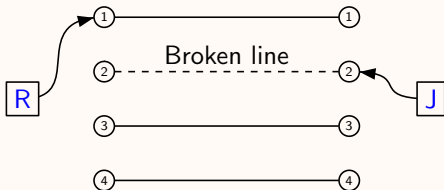
□ read-write action

○ read-only action

Example

Romeo and Juliet against the environment

- ▶ Want to communicate through the same communication line
- ▶ **At any time, one line is broken**
- ▶ **Environment** looks where R&J are connected, and then, atomically, possibly changes the broken line
- ▶ **Romeo/Juliet** looks status of lines and, atomically, chooses where to connect.
- ▶ Dependence between moves $R - E - J$
- ▶ If no fairness assumption: **E** can monopolize the game breaking lines forever



Communication architecture

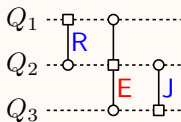
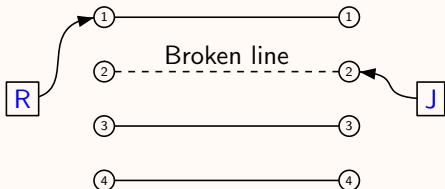
□ read-write action

○ read-only action

Example

Romeo and Juliet against the environment

- ▶ Want to communicate through the same communication line
- ▶ At any time, one line is broken
- ▶ Environment looks where R&J are connected, and then, atomically, possibly changes the broken line
- ▶ Romeo/Juliet looks status of lines and, atomically, chooses where to connect.
- ▶ Dependence between moves $R - E - J$
- ▶ If no fairness assumption: E can monopolize the game breaking lines forever



Communication architecture

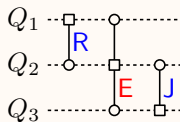
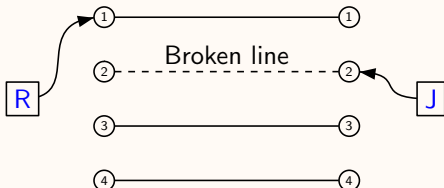
□ read-write action

○ read-only action

Example

Romeo and Juliet against the environment

- ▶ Want to communicate through the same communication line
- ▶ At any time, one line is broken
- ▶ **Environment** looks where R&J are connected, and then, atomically, possibly changes the broken line
- ▶ **Romeo/Juliet** looks status of lines and, atomically, chooses where to connect.
- ▶ Dependence between moves $R - E - J$
- ▶ If no fairness assumption: **E** can monopolize the game breaking lines forever



Communication architecture

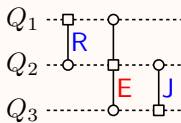
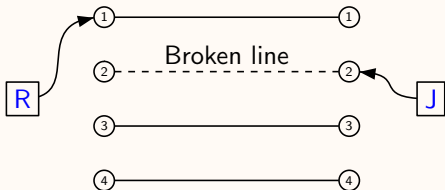
□ read-write action

○ read-only action

Example

Romeo and Juliet against the environment

- ▶ Want to communicate through the same communication line
- ▶ At any time, one line is broken
- ▶ **Environment** looks where R&J are connected, and then, atomically, possibly changes the broken line
- ▶ **Romeo/Juliet** looks status of lines and, atomically, chooses where to connect.
- ▶ **Dependence between moves R - E - J**
- ▶ If no fairness assumption: **E** can monopolize the game breaking lines forever



Communication architecture

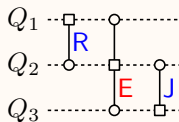
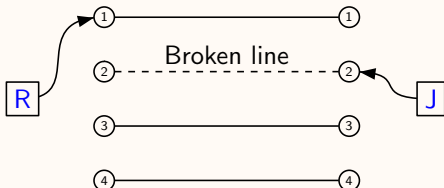
□ read-write action

○ read-only action

Example

Romeo and Juliet against the environment

- ▶ Want to communicate through the same communication line
- ▶ At any time, one line is broken
- ▶ **Environment** looks where R&J are connected, and then, atomically, possibly changes the broken line
- ▶ **Romeo/Juliet** looks status of lines and, atomically, chooses where to connect.
- ▶ Dependence between moves **R - E - J**
- ▶ **If no fairness assumption: E can monopolize the game breaking lines forever**



Communication architecture

□ read-write action

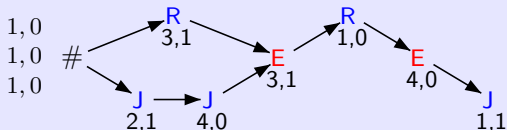
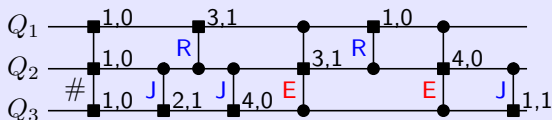
○ read-only action

Example (continued)

Moves and plays

- ▶ Fairness: **E** loses if it does not let **R&J** play $2k + 1$ times between 2 of its moves
- ▶ States of process i : $Q_i = \{1, 2, 3, 4\} \times \{0, 1\}$
- ▶ State $(3, 1) \in Q_1$ means **R** sits on line **3** and played an odd nb. of times
- ▶ State $(2, 0) \in Q_2$ line **2** is broken, and **E** played an even nb. of times
- ▶ Example of move: **E** reads $(3,1)$, $(1,0)$, $(4,0)$ on processes 1, 2 and 3 and writes $(3,1)$ on process 2.

A distributed play of the asynchronous system, **R&J** against **E**



Recap: architectures

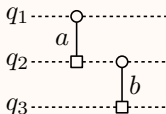
- ▶ \mathcal{P} : set of processes
- ▶ The alphabet Σ is partitioned in 2 teams Σ_0 and Σ_1
- ▶ Each player a of Σ atomically
 - ▶ reads states of processes from a given set $R(a) \subseteq \mathcal{P}$.
 - ▶ changes the state values for processes of $W(a) \subseteq \mathcal{P}$.

$$\forall a \in \Sigma, \quad \emptyset \neq W(a) \subseteq R(a)$$

$$\forall a, b \in \Sigma, \quad R(a) \cap W(b) \neq \emptyset \iff R(b) \cap W(a) \neq \emptyset$$

- ▶ **Dependence:** $a D b \iff R(a) \cap W(b) \neq \emptyset \iff R(b) \cap W(a) \neq \emptyset$
- ▶ Moves given by local transition functions $\delta_a \subseteq \prod_{i \in R(a)} Q_i \times \prod_{i \in W(a)} Q_i$

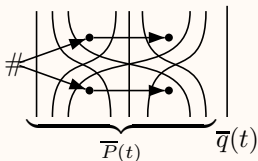
A “forbidden”, one-way architecture



Winning conditions

Controlled reachability conditions

- ▶ $\bar{q}(t)$: global state reached on t .
- ▶ $\bar{P}(t)$: set of global states seen along (finite) prefixes of play t



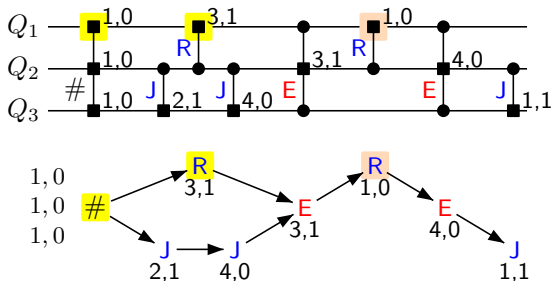
- ▶ Controlled reachability condition: $\mathcal{F} \subseteq 2^Q \times (Q \uplus \{\infty\})$
- ▶ A play t is winning if $(\bar{P}, \bar{q})(t) \in \mathcal{F}$
- ▶ Particular cases: reachability, safety
- ▶ Example $\text{Reach}(X)$ correspond to

$$\mathcal{F} = \left\{ (\bar{P}, \bar{q}) \mid X \cap (\bar{P} \cup \bar{q}) \neq \emptyset \right\} \cup \left\{ (\bar{P}, \infty) \mid X \cap \bar{P} \neq \emptyset \right\}$$

Causal vs. local memory

Memory

- ▶ Intuitively, maximal memory of a player is = history of the play he can observe
- ▶ Local memory: a player remembers the write actions.



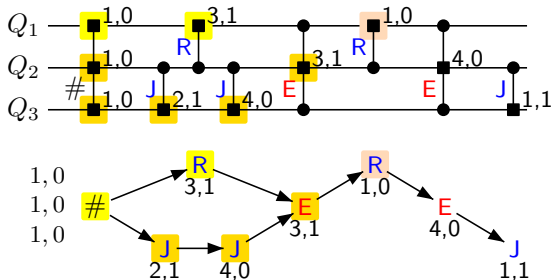
Causal memory

- ▶ Causal memory: players gather and forward as much information as possible.
- ▶ The strategies in asynch. distributed games can use an abstraction of causal memory
- ▶ Due to symmetry condition: causal view is implementable.

Causal vs. local memory

Memory

- ▶ Intuitively, maximal memory of a player is = history of the play he can observe
- ▶ Local memory: a player remembers the write actions.



Causal memory

- ▶ Causal memory: players gather and forward as much information as possible.
- ▶ The strategies in asynch. distributed games can use an abstraction of causal memory
- ▶ Due to symmetry condition: causal view is implementable.

Summary

The main result (again)

Asynchronous distributed games with controlled reachability conditions
are decidable
for symmetric series-parallel systems and causal memory strategies.

Remarks

- ▶ Both teams have to participate in the computation of the causal memory
- ▶ Even reachability conditions may require arbitrary large memory (on series-parallel architectures)
- ▶ Natural candidates (e.g., second approximation) do not seem to be suited

The important point

Love wins, Romeo and Juliet have a strategy to meet.

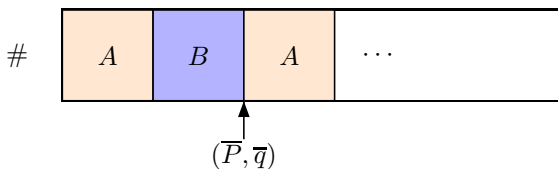
Series-parallel systems and proof techniques

Cograph dependence alphabets: smallest family of dep. alphabets st.

- ▶ Every singleton is a cograph,
- ▶ If $A \cap B = \emptyset$, and (A, D_A) , (B, D_B) cographs, then so are $(A \cup B, D_A \cup D_B)$ and $(A \cup B, D_A \cup D_B \cup A \times B \cup B \times A)$

Induction on Σ : Σ_0 has a winning strategy, \Rightarrow it has a **small** winning strategy

Difficult case:



- ▶ From a play, recover small plays on A and B and associated strategies.
- ▶ By induction, replace by **small** strategies
- ▶ Main problem: team 0 has to identify on which small play it is acting.
- ▶ Team 0 has to compute in a distributed way the corresponding (\bar{P}, \bar{q}) .