

Shared-Memory Systems and Charts

Rémi MORIN

Université de la Méditerranée
Laboratoire d'Informatique Fondamentale de Marseille

Peterson's mutual exclusion protocol

Process 1

repeat:

f[1] ← true;

Turn ← 2;

**wait (f[2] = false
 or Turn = 1);**

Critical Section(1);

f[1] ← false;

Process 2

repeat:

f[2] ← true;

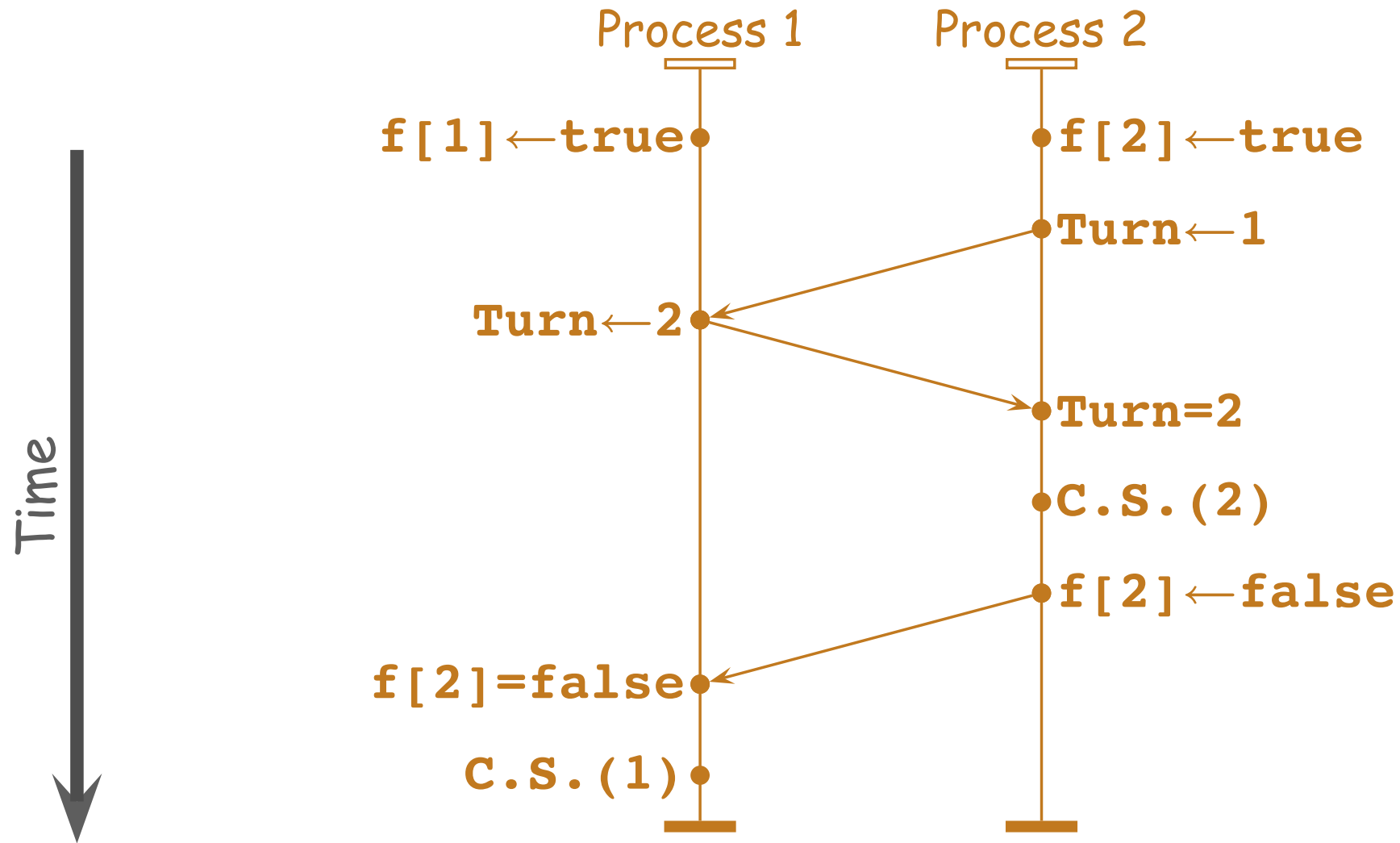
Turn ← 1;

**wait (f[1] = false
 or Turn = 2);**

Critical Section(2);

f[2] ← false;

Executions as partial orders (pomsets)



✓ Foreword



Model and semantics

Expressive power & MSO logic

Specifications with automata

Checking SMC specifications

A simple model for shared-memory systems

Let Σ be a fixed alphabet.

A **shared-memory system** consists of

- a set of **registers** \mathcal{R} , a set of **data** \mathcal{D} ,
- a set of **processes** \mathcal{P}
- for each action $a \in \Sigma$: a **non-empty** subset $\text{Loc}(a) \subseteq \mathcal{P}$
- an initial **configuration** $i \in \mathcal{Q}$
- some final **configurations** $F \subseteq \mathcal{Q}$.
- and for each action $a \in \Sigma$: a set of **rules** Δ_a

• A **configuration** is a mapping $q : \mathcal{R} \rightarrow \mathcal{D}$

• A **rule** is a triple $\rho = (v, a, v')$ where $v, v' : \mathcal{R} \rightarrow \mathcal{D}$

Guard

Update

Sequential operational semantics

For any two states $q, q' \in Q$ and any rule $\rho = (v, a, v') \in \Delta_a$, we denote by

- $a_\rho = a$ the action performed by ρ
- $R_\rho = \text{dom}(v)$ the subset of registers read by ρ
- $W_\rho = \text{dom}(v')$ the subset of registers modified by ρ

We put $q \xrightarrow{\rho} q'$ if

- $q|_{R_\rho} = v$ (the rule is **enabled** in q)
- $q'|_{W_\rho} = v'$ (the rule is **applied** in q')
- $q'(r) = q(r)$ for all $r \in \mathcal{R} \setminus W_\rho$ (nothing else happens inbetween)

Special case [Zielonka, RAIRO, 1987]

An **asynchronous automaton** is an SMS such that

- $\mathcal{P} = \mathcal{R}$ and
- for all rules $\rho \in \Delta_a$, $R_\rho = \text{Loc}(a) = W_\rho$.

May-Occur-Concurrently relation

Let $\rho, \rho' \in \Delta$ be two rules.

We put $\rho \parallel \rho'$ if

- $\text{Loc}(a_\rho) \cap \text{Loc}(a_{\rho'}) = \emptyset$,
- $W_\rho \cap (R_{\rho'} \cup W_{\rho'}) = \emptyset$ and
- $W_{\rho'} \cap (R_\rho \cup W_\rho) = \emptyset$.

Intuitively, two rules may occur concurrently if they correspond to actions occurring on disjoint sets of processes and if each rule does not modify the registers read or written by the other.

Partial-order semantics (1/2)

Let $t = (E, \preceq, n)$ be a labeled partial order, i.e. a *partially ordered multiset* (for short: a pomset) over Σ .

A *run* of t is a mapping $\rho : E \rightarrow \Delta$ such that

R_0 : For all $e \in E$, $a_{\rho(e)} = n(e)$

(rule action matches event action)

R_1 : For all $e_1, e_2 \in E$ with $\rho(e_1) \not\parallel \rho(e_2)$, $e_1 \preceq e_2$ or $e_2 \preceq e_1$

(dependent rules cannot occur concurrently)

R_2 : For all $e_1, e_2 \in E$ with $e_1 \text{---} \prec e_2$, $\rho(e_1) \not\parallel \rho(e_2)$

(waiting means rule dependency)

where $x \text{---} \prec y$ means: $x \prec y$ and $x \prec z \preceq y$ implies $z = y$.

Partial-order semantics (2/2)

Let H be a downward-closed subset of events (a prefix of t).
The *configuration* $q_{\rho,H}$ reached after H with run ρ is such that


$$q_{\rho,H}(r) = \begin{cases} v'_{\rho(e)}(r) & \text{if } e = \max\{f \in H \mid r \in W_{\rho(e)}\} \\ q_{\rho,H}(r) = 1(r) & \text{if there is no such event} \end{cases}$$

A run ρ of t is *applicable* if the rule $\rho(e)$ is enabled in $q_{\rho, \downarrow e \setminus \{e\}}$ for all events $e \in E$.

An applicable run of $t = (E, \preceq, \eta)$ is *accepting* if $q_{\rho,E} \in F$.

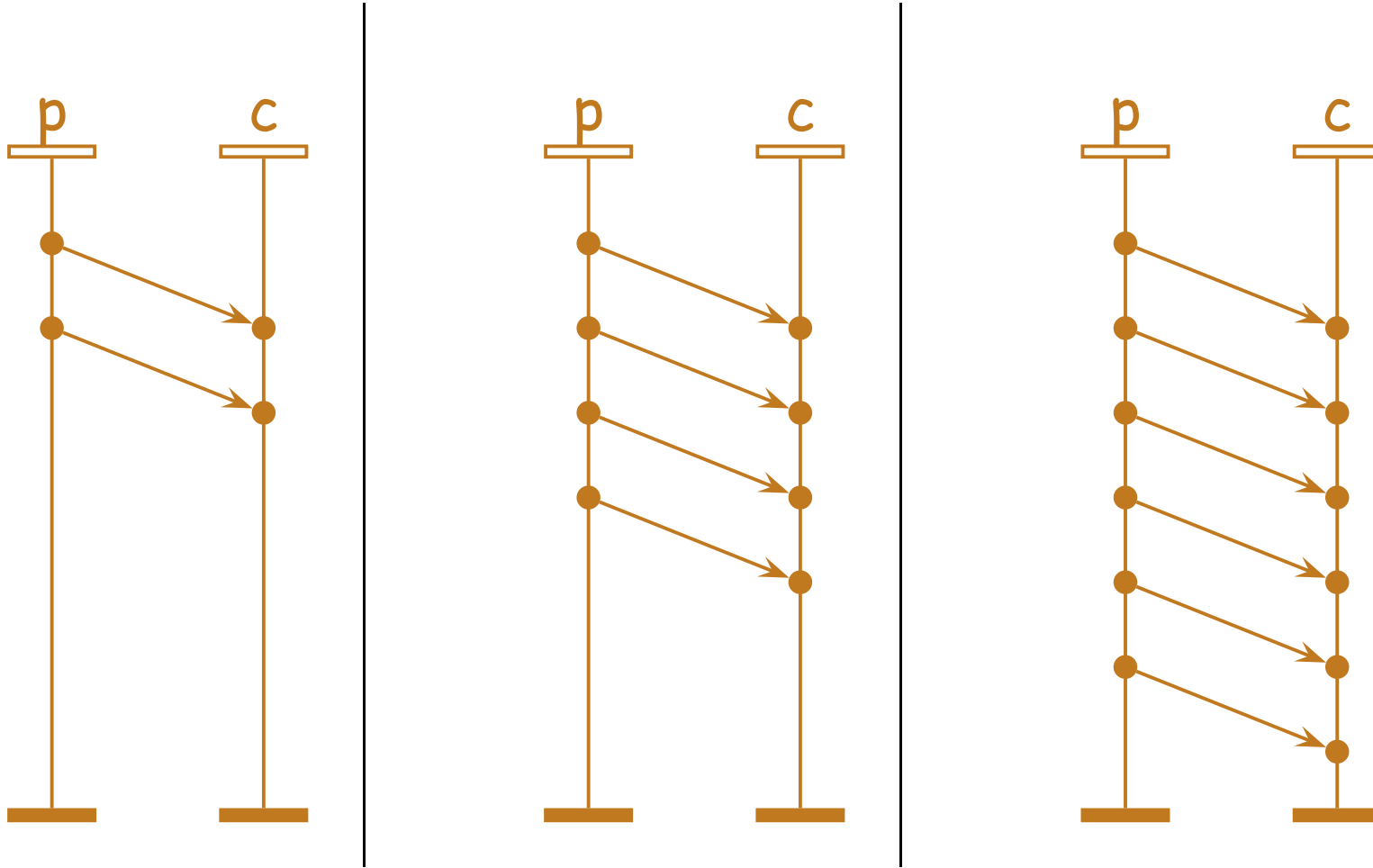
Definition

The language $\mathcal{L}(\mathcal{S})$ recognized by \mathcal{S} collects all pomsets which admit some accepting run.

- ✓ Foreword
- ✓ Model and semantics
-  Expressive power & MSO logic
Specifications with automata
Checking SMC specifications

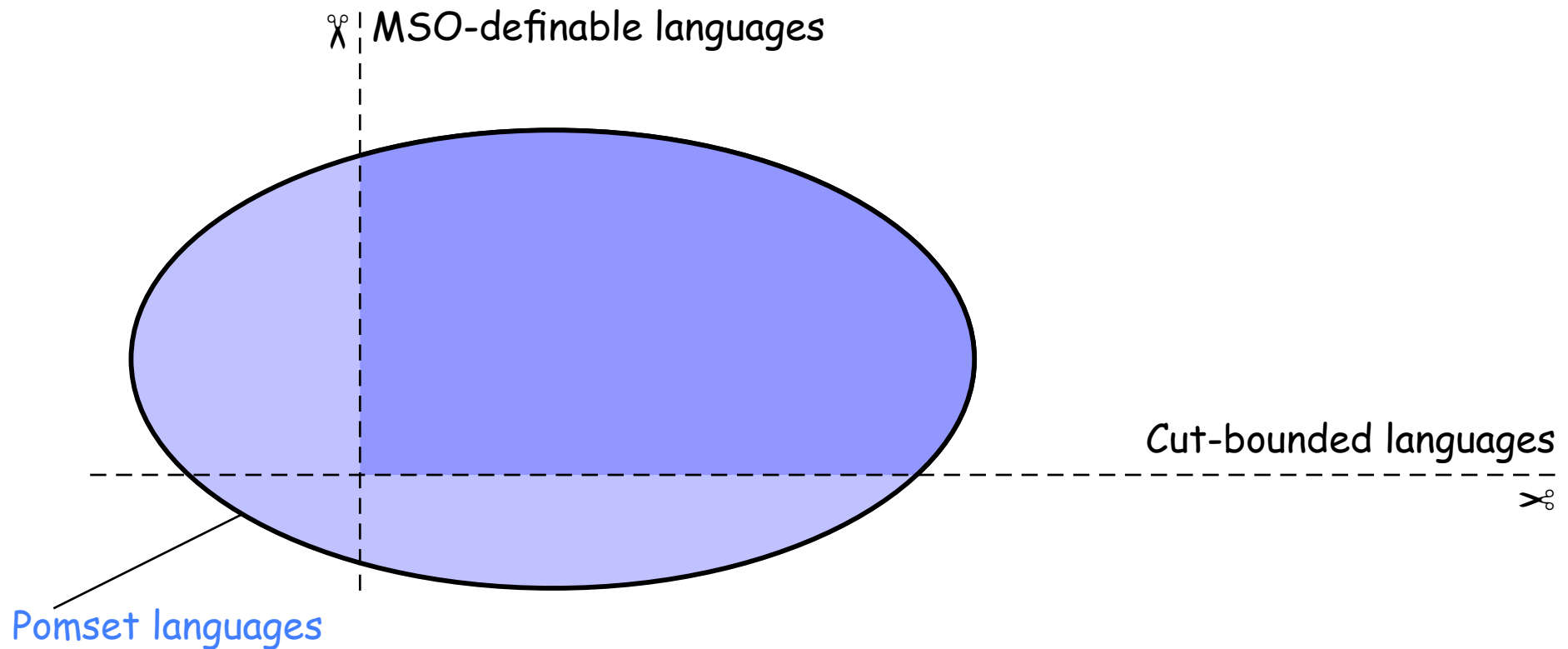
Question!

Let $\Sigma = \{p, c\}$ and \mathcal{L} be the set of all ladders.



Does any SMS recognize this language?

First result



Theorem (Expressive power of shared-memory systems)

A pomset language is recognized by some **finite SMS** iff it is MSO-definable and *cut-bounded*.

MSO logic

The language of all ladders is MSO-definable by the conjunction of the following sentences:

$$\forall y : P_c(y) \rightarrow \exists x.(P_p(x) \wedge x \prec y)$$

$$\forall x, z : P_p(x) \rightarrow \exists y.(P_c(y) \wedge x \prec y)$$

$$\forall x, y : (P_p(y) \wedge x \preceq y) \rightarrow P_p(x)$$

Cut-bounded languages

The (universal) *cut-width* of $t = (E, \preceq, n)$ is

$$CW(t) = \max_{H \text{ prefix of } t} \#\{ (h, e) \in H \times (E \setminus H) \mid h \preceq e \}$$

Definition

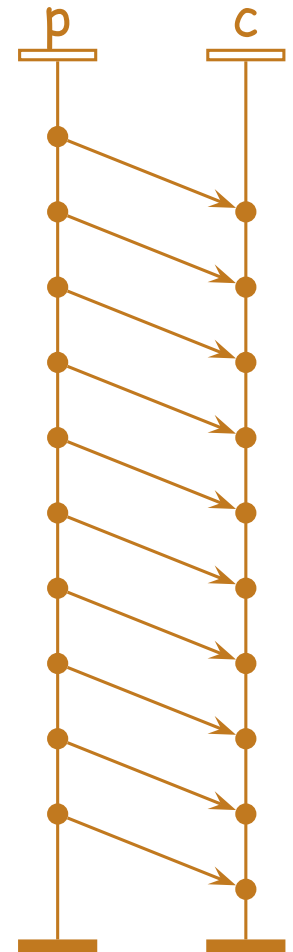
The *cut-bound* $B \in \mathbb{N} \cup \{\infty\}$ of \mathcal{L} is

$$\sup\{ CW(t) \mid t \in \mathcal{L} \}$$

\mathcal{L} is *cut-bounded* if its cut-bound is $< \infty$.

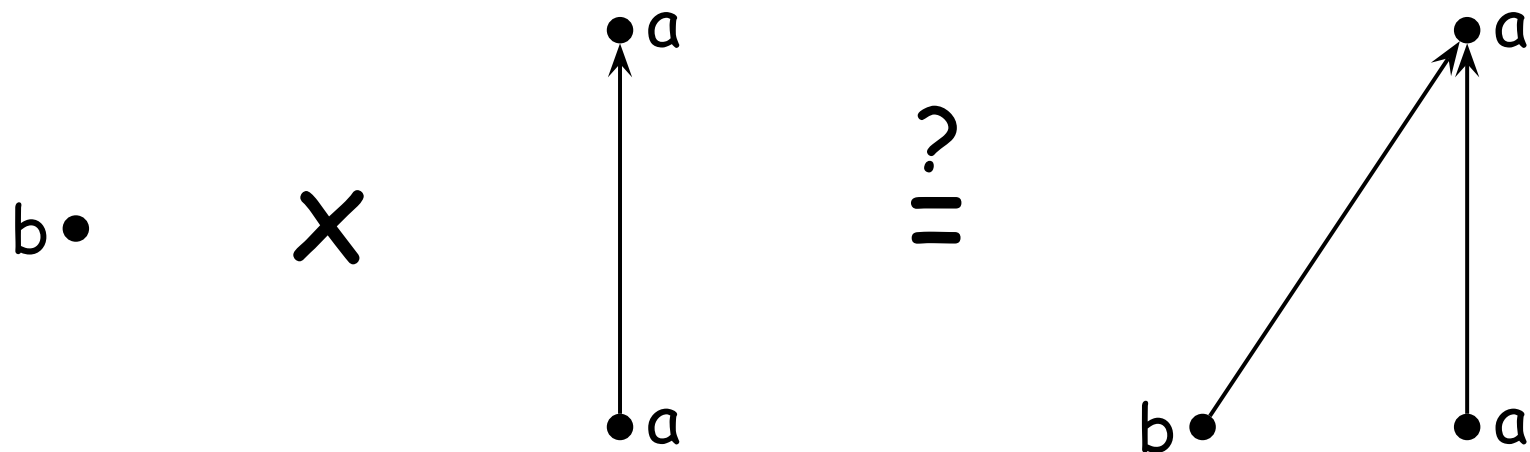
Example

The language of all ladders is **not** cut-bounded.



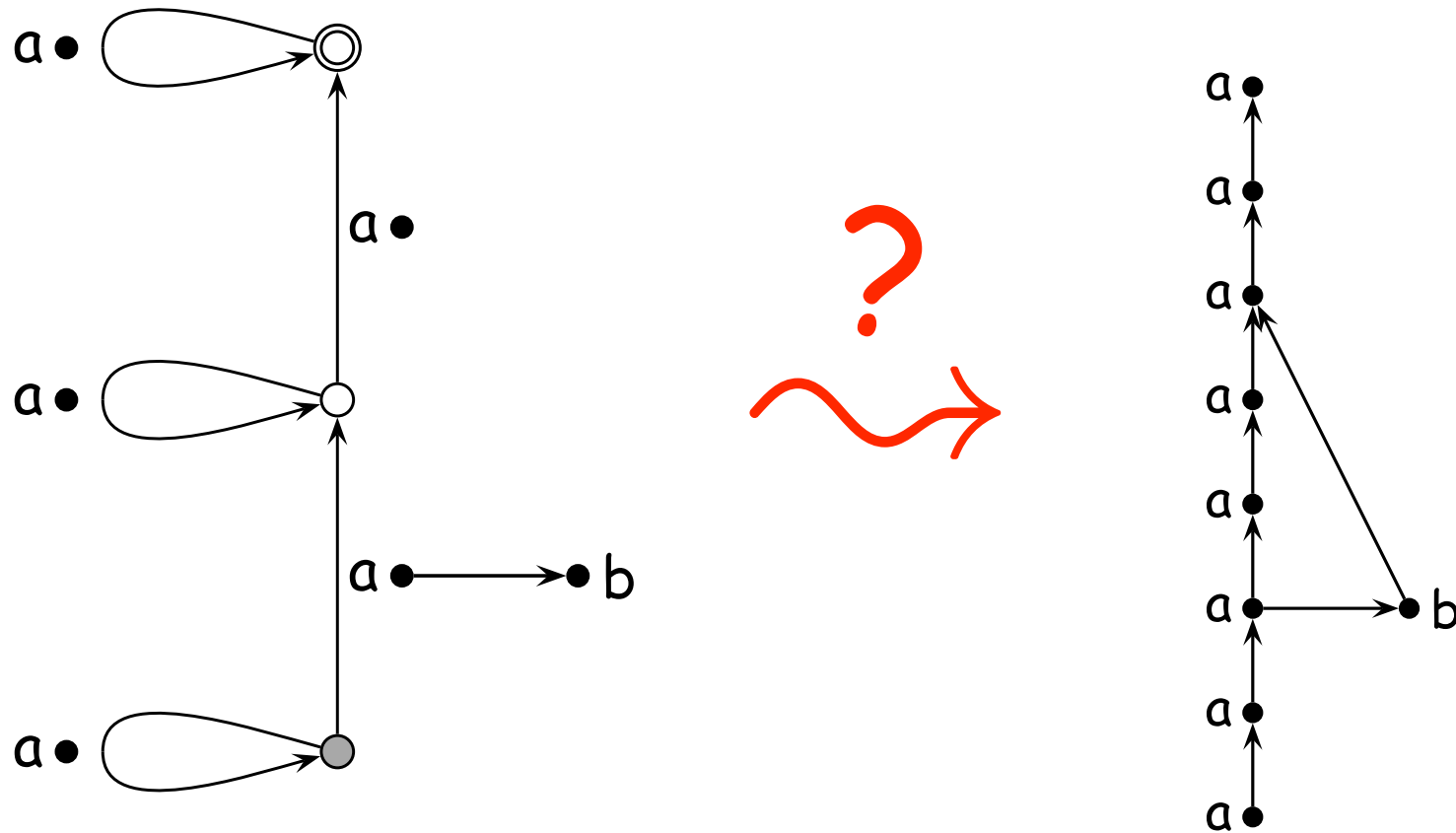
- ✓ Foreword
- ✓ Model and semantics
- ✓ Expressive power & MSO logic
- ☞ Specifications with automata
- Checking SMC specifications

How to concatenate two pomsets?



We have to distinguish between a's

How to concatenate two pomsets?



We have to distinguish between a's

Pomsets with gates

Let G be a finite and non-empty set of *gates*.

We consider the extended alphabet $\Gamma = \Sigma \times 2^G \setminus \{\emptyset\}$.

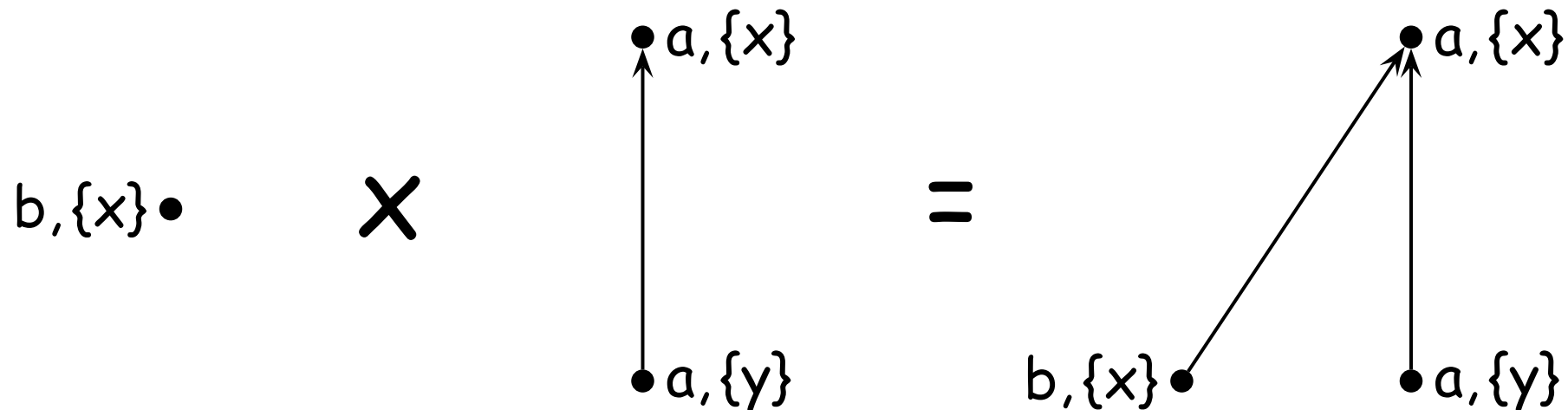
We put $(a, H) \parallel_{\Gamma} (a', H')$ if $H \cap H' \neq \emptyset$ or $a = a'$.

Definition (Pomsets with gates)

A *shared-memory chart* (an SMC) is a pomset $t = (E, \preceq, n)$ over Γ such that we have either $e_1 \preceq e_2$ or $e_2 \preceq e_1$ for any two events e_1 and e_2 with $n(e_1) \parallel_{\Gamma} n(e_2)$.

We denote by SMC the set of all SMCs.

Product of SMCs

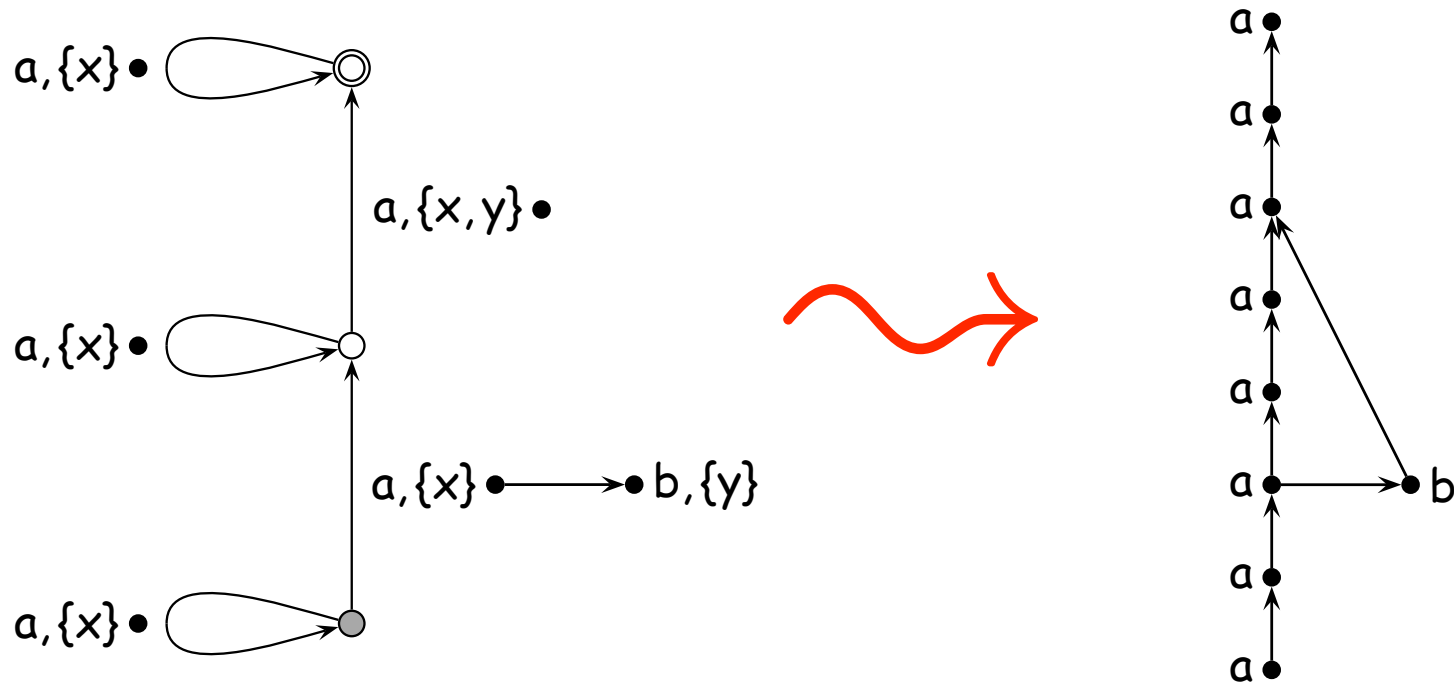


Definition (Product of pomsets with gates)

Given two SMCs $t_1 = (E_1, \preceq_1, \eta_1)$ and $t_2 = (E_2, \preceq_2, \eta_2)$ the asynchronous product $t_1 \cdot t_2$ is the pomset $t = (E, \preceq, \eta)$ where $E = E_1 \cup E_2$, $\eta = \eta_1 \cup \eta_2$, and \preceq is the transitive closure of


$$\preceq_1 \cup \preceq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid \eta(e_1) \parallel_{\Gamma} \eta(e_2)\}$$

Rational SMC languages



Definition (Automata over pomsets with gates)

An *SMC specification* is an automaton $\mathcal{A} = (Q, i, \rightarrow, F)$ where Q is a finite set of states, with initial state i , $\rightarrow \subseteq Q \times \text{SMC} \times Q$ is a finite set of transitions labeled by SMCs, and $F \subseteq Q$ is a subset of final states.

- ✓ Foreword
- ✓ Model and semantics
- ✓ Expressive power & MSO logic
- ✓ Specifications with automata
-  Checking SMC specifications

How to detect unbounded specifications?

Definition

Let $t = (E, \preceq, n)$ be an SMC. The *communication graph* of t is the directed graph $CG(t) = (V, \rightarrow)$ over the set $V = \bigcup_{e \in E} \pi_2(n(e))$ of active gates in t such that $g \rightarrow g'$ if there are $e, e' \in E$ for which $g \in \pi_2(n(e))$, $g' \in \pi_2(n(e'))$ and

- either $n(e) \not\parallel_{\Gamma} n(e')$
- or $e \prec e'$.

Checking unboundedness

Theorem

The pomset language $\mathcal{L}_\Sigma(\mathcal{A})$ of an SMC specification \mathcal{A} is cut-bounded iff for any loop $q_0 \xrightarrow{t_1} \dots \xrightarrow{t_n} q_n = q_0$, all connected components of the communication graph $CG(t_1 \cdot \dots \cdot t_n)$ are strongly connected.

Consequently checking for cut-boundedness of a given SMC specification is decidable. It is actually easy to show that this problem is co-NP-complete.

How to detect non-implementable specifications?

We cannot decide whether an SMC specification describes an implementable language, since this question is already undecidable for Mazurkiewicz traces.

Definition

An SMC specification is *loop-connected* if for all loops $q_0 \xrightarrow{t_1} \dots \xrightarrow{t_n} q_n = q_0$ the communication graph of the SMC $t_1 \cdot \dots \cdot t_n$ is connected.

Theorem

A cut-bounded language is MSO-definable if and only if it is the language of a loop-connected SMC specification.

Conclusion

- We have presented a characterization of the expressive power of shared-memory systems
 1. in terms of **logic definability** and **cut-boundedness**
 2. in terms of **automata** over pomsets with **gates**.
- This model of concurrency and this algebraic framework **generalize** the theory of Mazurkiewicz traces and message sequence charts.
- These results should be extended soon to systems *with autoconcurrency*.
- A *simpler* notion of **communication graph** may be designed.

Questions?

Unambiguity & determinism

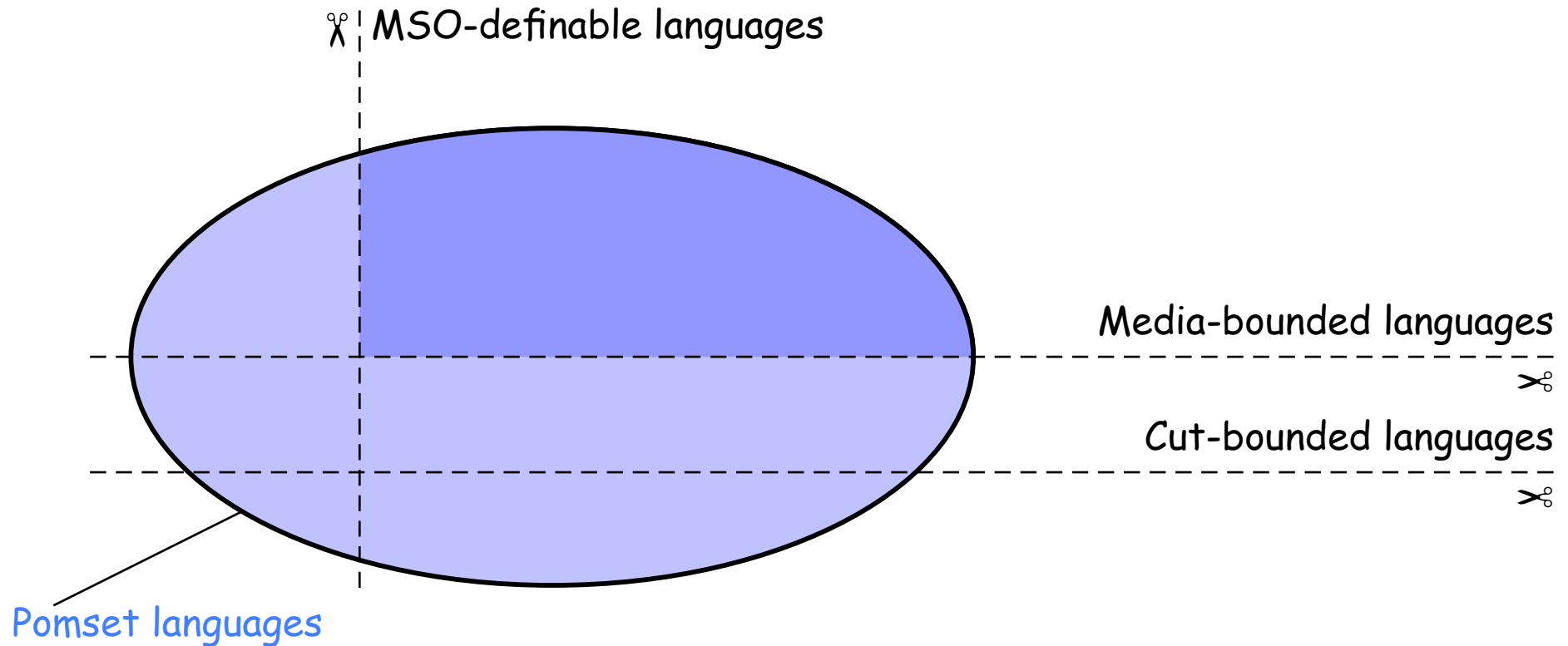
An SMS is *unambiguous* if each pomset admits at most one applicable run.

An SMS is *deterministic* if for each $a \in \Sigma$ and each reachable configuration q , there exists *at most one* rule $\rho \in \Delta_a$ such that $q \xrightarrow{\rho} q'$.

Clearly:

Any deterministic SMS is unambiguous.

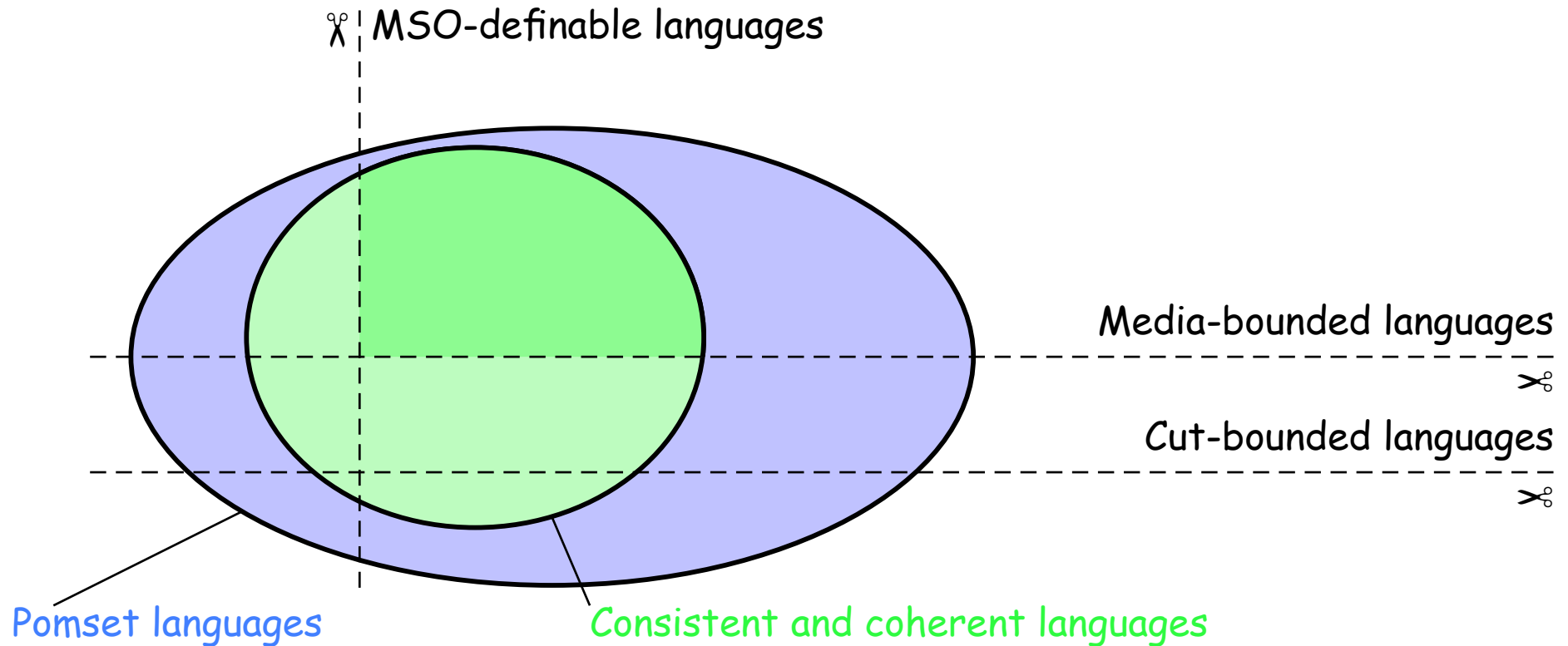
Unambiguous case



Theorem \simeq [M., IJFCS, 2010]

A pomset language is recognized by some *unambiguous* finite SMS iff it is MSO-definable and *media-bounded*.

Deterministic case



Theorem \simeq [M., CONCUR'08]

A pomset language is recognized by some *deterministic* finite SMS iff it is MSO-definable, media-bounded, *coherent* and *consistent*.

SMCs vs. Mazurkiewicz traces and MSCs

Any *Mazurkiewicz trace* can be regarded as an SMC where each action is a gate and each event labeled by a is associated with the set of actions dependent with a .

Similarly any *message sequence chart* can be regarded as an SMC where gates are processes and each event is associated with the (single) process where it occurs.

Moreover these identifications preserve the *product* of traces and MSCs

In that way, SMCs appear as a formal generalization of both Mazurkiewicz traces and message sequence charts.