A look into distributed control

Anca Muscholl

LaBRI, Bordeaux

Janvier 2011

- Sequential systems

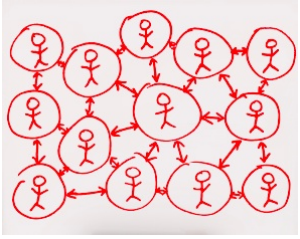
  - well-understood analysis
    (model-checking, controller synthesis,..)
  - powerful automata-based verification techniques
  - manipulating "standard" objects
    (words, trees,..)

Motivation

- Sequential systems

  - well-understood analysis (model-checking, controller synthesis,..)
  - powerful automata-based verification techniques
  - manipulating "standard" objects (words, trees,..)

- Asynchronous systems

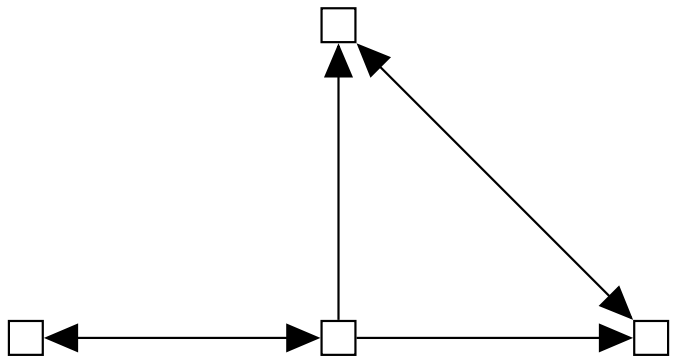


  - analysis more difficult
  - automata-based methods harder to obtain
  - manipulated objects depend on the model

Asynchronous systems

Models

Processes with links. A process is an automaton (e.g. finite-state).



Links as channels

Links are channels and processes have send and receive operations:
communicating automata, message sequence charts. Turing powerful.

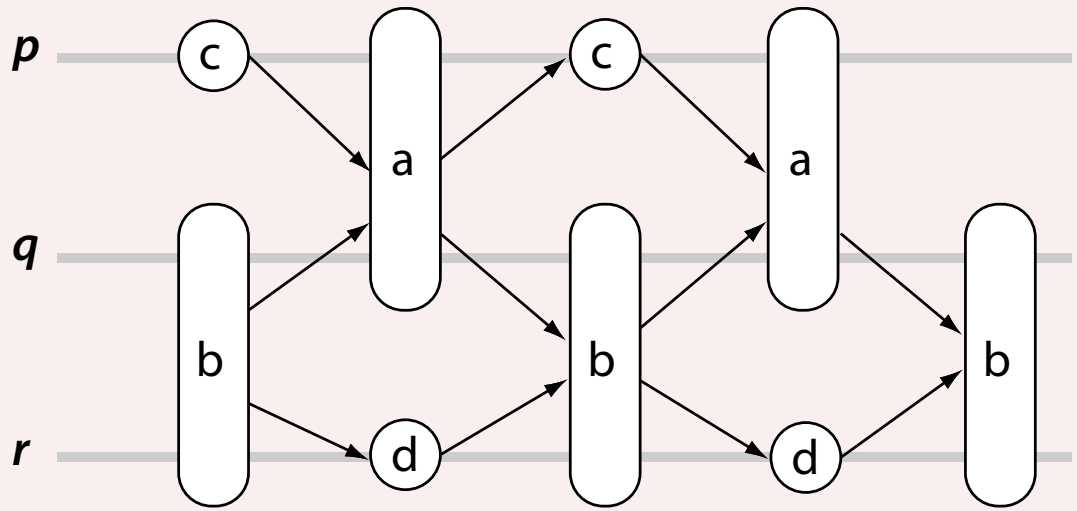
Links as synchronization

Links are shared variables and processes can read/write:
asynchronous automata, Mazurkiewicz traces, event structures.
Regular languages.

Control for shared variables

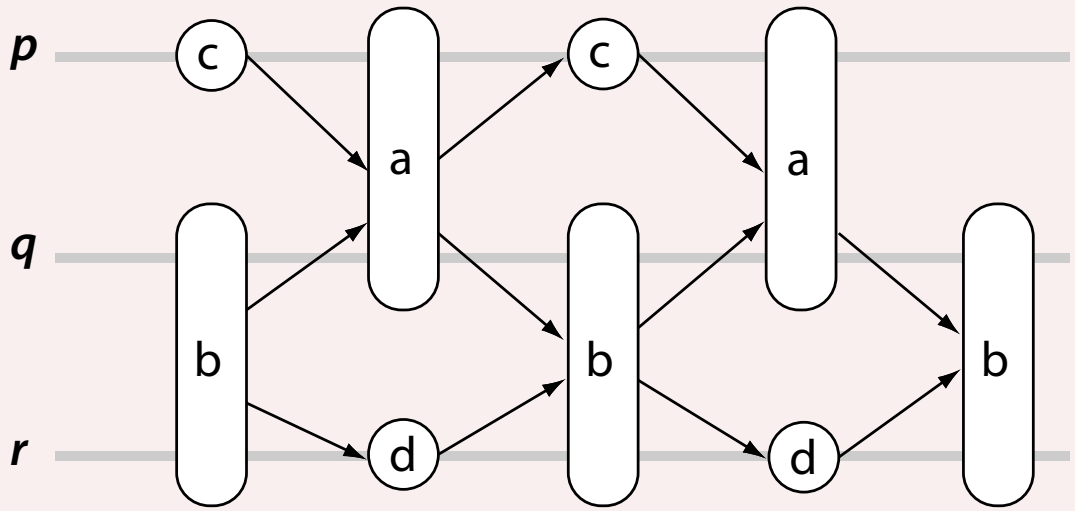
MODEL

Mazurkiewicz traces



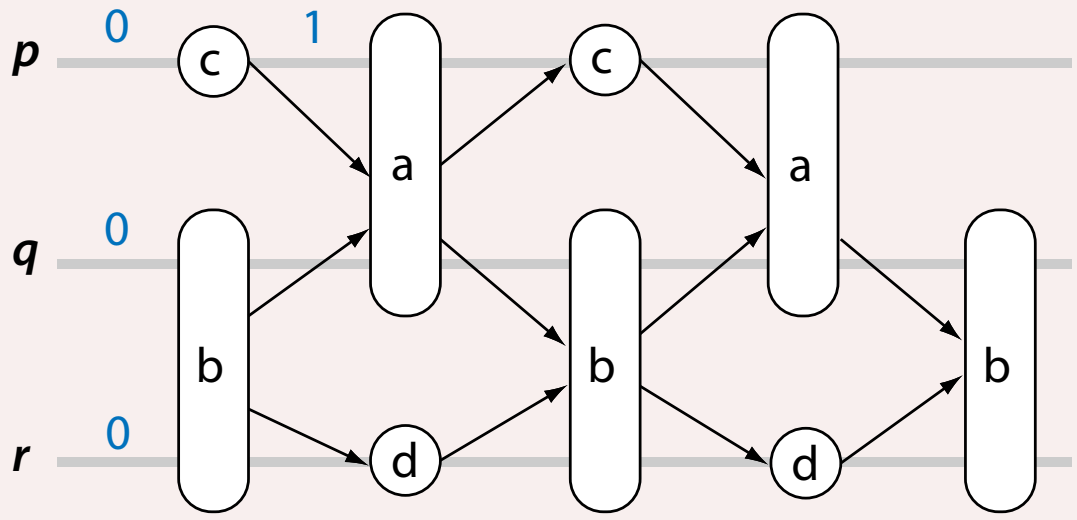
- $dom : \Sigma \to (2^{\mathbb{P}} \setminus \emptyset)$: distribution of actions over processes.
- Dependence relation: $(a, b) \in D$ if $dom(a) \cap dom(b) \neq \emptyset$.
  Independence $I = (\Sigma \times \Sigma) \setminus D$.
- Trace $[bcadbcadb] = \{bcadbcadb, cbadbcdab, \ldots\}$.

Automata



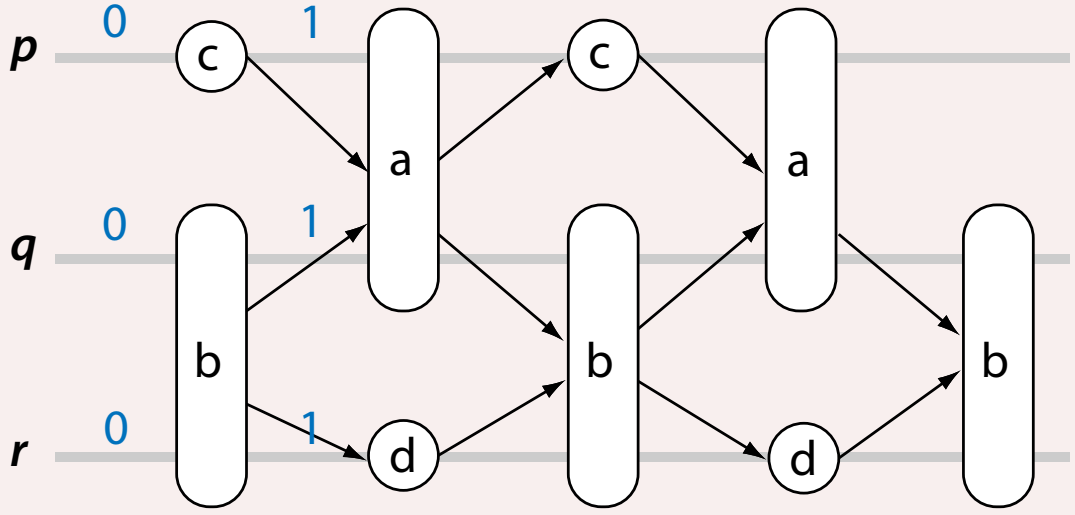
- Local states sets S_p, S_q, S_r.
- Local transitions $\delta_b : S_q \times S_r \to S_q \times S_r$. Letter b reads/writes its domain $dom(b) = \{q, r\}$.

Automata



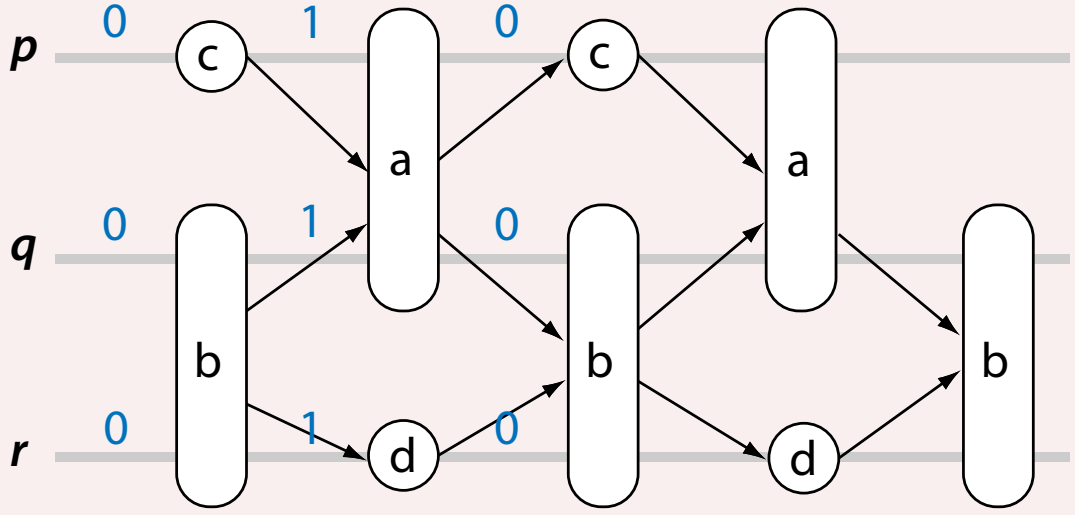
- Local states sets S_p, S_q, S_r.
- Local transitions $\delta_b : S_q \times S_r \to S_q \times S_r$. Letter b reads/writes its domain $dom(b) = \{q, r\}$.

Automata



- Local states sets S_p, S_q, S_r.
- Local transitions $\delta_b : S_q \times S_r \to S_q \times S_r$. Letter b reads/writes its domain $dom(b) = \{q, r\}$.

Automata



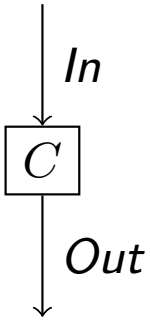
- Local states sets S_p, S_q, S_r.
- Local transitions $\delta_b : S_q \times S_r \to S_q \times S_r$. Letter b reads/writes its domain $dom(b) = \{q, r\}$.
- Asynchronous automata accept trace-closed languages.

Automata

- Given a distributed alphabet and a regular (trace-closed) language L, one can construct a deterministic asynchronous automaton for L [Zielonka'89].
- Complexity: polynomial in DFA for L, exponential in $\#$ processes [Genest et al.'10].

CONTROL

The synthesis/control problem

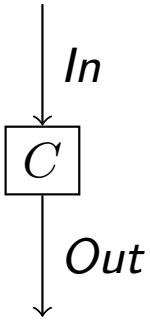


$S \subseteq (\mathit{In}\ ;\ \mathit{Out})^*$

Centralized control (Church)

- Given: specification S.
- Output: finite automaton (controller) C with $C \subseteq S$ + additional requirements (e.g., unconstrained inputs).
- Decidable: tree automata.

The synthesis/control problem



$S \subseteq (\mathit{In}\ ;\ \mathit{Out})^*$

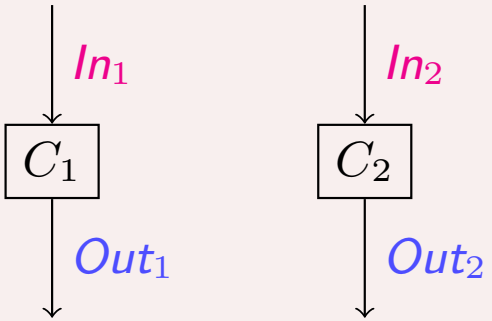
Centralized control (Church)

- Given: specification S.
- Output: finite automaton (controller) C with $C \subseteq S$ + additional requirements (e.g., unconstrained inputs).
- Decidable: tree automata.

Distributed control

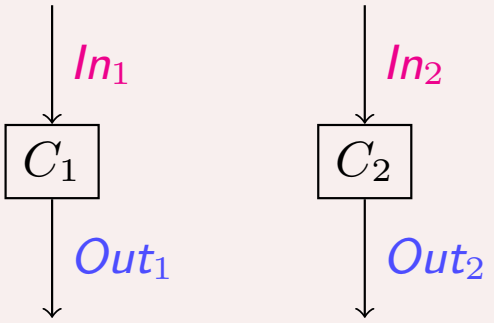
- Comes along with a distributed architecture (e.g., distributed alphabet).
- In general undecidable [Peterson/Reif, Pnueli/Rosner].
- Important: use adequate specifications (e.g. trace-closed ones for asynchronous automata).

Undecidability



- Partial information: each process knows only its input. Specification talks about In/Out of both processes.
- Specification is not trace closed.

Undecidability



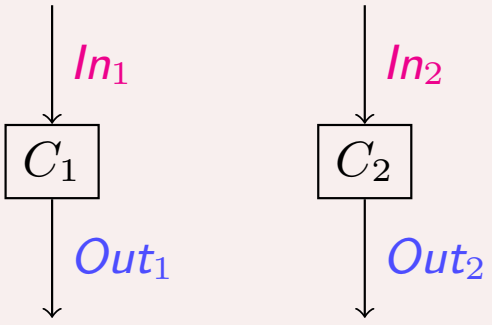
$a_1 - 0 - b_1 - 1 \cdots$

$a_2 - 1 - b_2 - 0 \cdots$

- Partial information: each process knows only its input. Specification talks about *In*/*Out* of both processes.
- Specification is not trace closed.

$a_1 a_2 01 b_1 b_2 10 \ldots + a_1 0 a_2 b_1 11 a_3 b_2 \ldots$

Undecidability



$a_1 - 0 - b_1 - 1 \cdots$

$a_2 - 1 - b_2 - 0 \cdots$

- Partial information: each process knows only its input. Specification talks about *In*/*Out* of both processes.
- Specification is not trace closed.

$a_1 a_2 01 b_1 b_2 10 \ldots + a_1 0 a_2 b_1 11 a_3 b_2 \ldots$

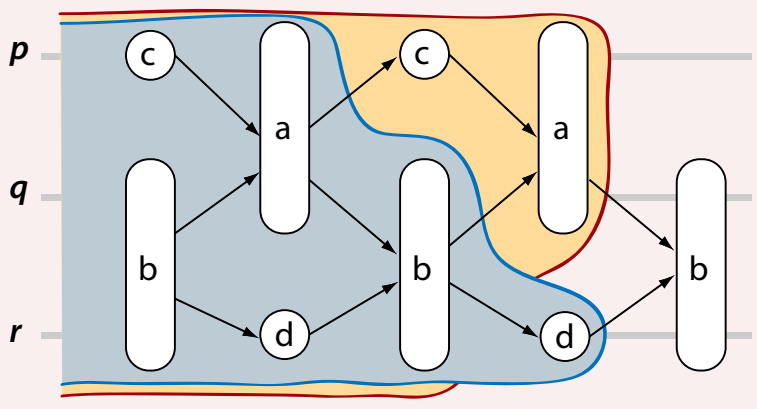
Pnueli & Rosner setting

- Processes behave synchronously, exchanging finite information over communication links. Links are either external (communication with environment) or internal.
- Only pipeline architectures are decidable [Pnueli/Rosner]. Various refinements of specifications [Madhusudan/Thiagarajan, Finkbeiner/Schewe, Gastin et al.].
- Proof technique for pipelines: tree automata.

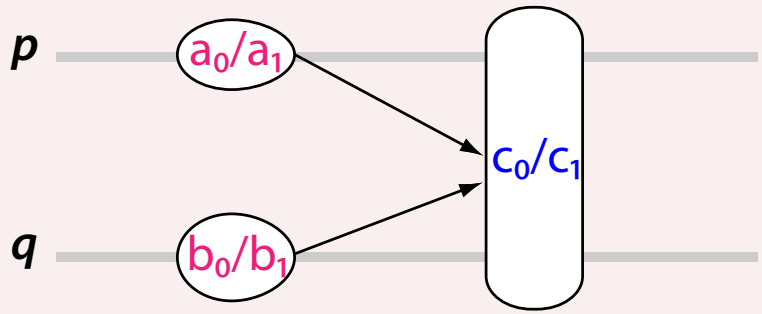
Formally

- Two kinds of actions: controllable (system) and uncontrollable (environment).
- Controllable actions are constrained such that the specification is respected (+ further requirements, e.g. deadlock-free).

Control strategies: causal past.
Unbounded information flow.



Example specifications



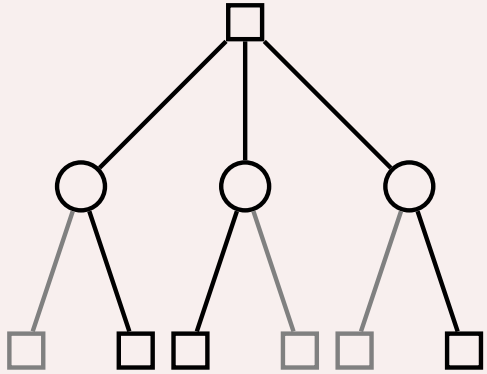
1. $a_i b_j c_k$ with $k = i$.
2. $a_i b_j c_k$ with $k = i \cdot j$.

Two methods of control

- Process-based [Madhusudan et al.]: Processes decide what actions they want to do.
- Action-based [Gastin et al.]: Actions decide whether they can execute.

Synchronous systems: trees

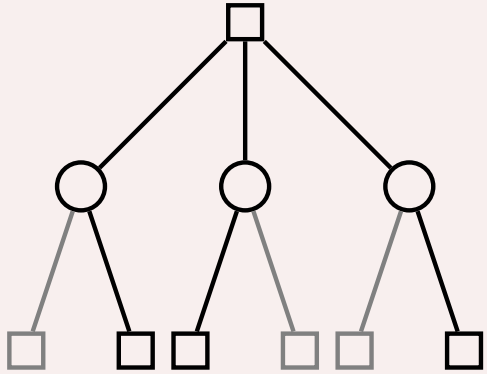
Synthesis considers tree properties (strategy trees, Rabin).



From trees to event structures

Synchronous systems: trees

Synthesis considers tree properties (strategy trees, Rabin).



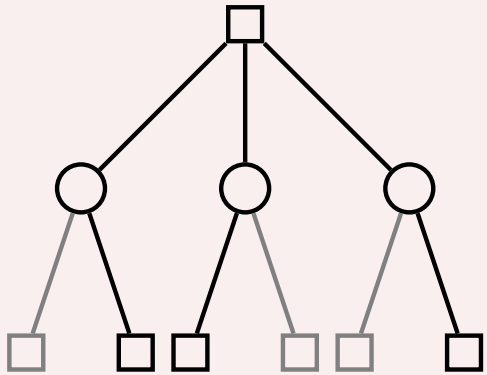
Asynchronous systems: event structures

Branching behavior of asynchronous systems: event structures.

From trees to event structures

Synchronous systems: trees

Synthesis considers tree properties (strategy trees, Rabin).



Asynchronous systems: event structures

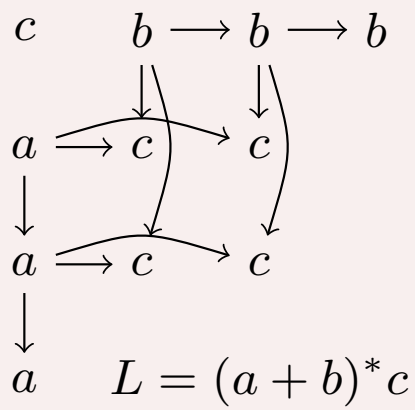
Branching behavior of asynchronous systems: event structures.

From traces to event structures

A prefix-closed trace language L defines a Σ-labeled event structure:

- Nodes: prime traces from L.
- Partial order: trace prefix relation.
- Conflict relation: no common extension.
- Label: maximal element of the trace.

$\Sigma = \{a, b, c\}$, $D: \ a - c - b$



$L = (a + b)^* c$

Solution?

The control problem for an asynchronous automaton $\mathcal{A}$ can be reduced to the satisfiability of a monadic second-order (MSO) formula over the event structure $ES(\mathcal{A})$ of $\mathcal{A}$ [Madhusudan et al.].

Solution?

The control problem for an asynchronous automaton $\mathcal{A}$ can be reduced to the satisfiability of a monadic second-order (MSO) formula over the event structure $ES(\mathcal{A})$ of $\mathcal{A}$ [Madhusudan et al.].

Bad news

- There exist asynchronous automata $\mathcal{A}$ s.t. $ES(\mathcal{A})$ has undecidable MSO theory.
- Decidability of MSO is not necessary for deciding the control problem.

Partial results

- Process-based control is decidable for asynchronous automata with strong synchronization ("there is no loop with a concurrent event") [Madhusudan et al.].
- Action-based control is decidable for asynchronous automata with co-graph dependence relation [Gastin et al.].
- Process-based control reduces to action-based control.

Control problem in asynchronous setting: open questions.

- Problem 1: decidability of the control problem?
- Problem 2: characterize asynchronous automata with decidable MSO theory.

  Thiagarajan's conjecture: $ES(\mathcal{A})$ has decidable MSO theory iff $\mathcal{A}$ has no parallel loops. Conjecture holds for co-graph dependence alphabets.
- Problem 3: Reduction from action-based control to process-based control?
- Problem 4: Control of communicating automata?

  For lossy communicating automata: monotonic games [Abdulla et al.]

Control problem in asynchronous setting: open questions.

- Problem 1: decidability of the control problem?
- Problem 2: characterize asynchronous automata with decidable MSO theory.

  Thiagarajan's conjecture: $ES(\mathcal{A})$ has decidable MSO theory iff $\mathcal{A}$ has no parallel loops. Conjecture holds for co-graph dependence alphabets.
- Problem 3: Reduction from action-based control to process-based control?
- Problem 4: Control of communicating automata?

  For lossy communicating automata: monotonic games [Abdulla et al.]

Merci !