Implementing Distributed Control on Star Architectures

Hugo Gimbert [1] Anca Muscholl [1] K Siddharth [2]

[1]LaBRI, University of Bordeaux I

[2]Chennai Mathematical Institute

Games 2012

Introduction

Synthesis

Given a specification, construct a “plant” that satisfies it (writing correct programs).

Control

More general: given a plant and a specification, construct a “controller” so that the plant + controller satisfies it.

We study control on Distributed Systems.

Distributed Models

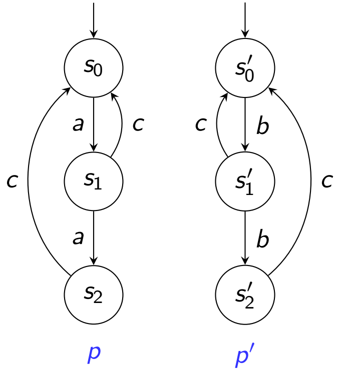
There is no canonical model. Many powerful ones, like communicating automata, but even the simplest problems (eg. reachability) are undecidable.

We consider Zielonka or Asynchronous automata.

Why Zielonka Automata?

- Closely linked to Mazurkiewicz traces - have rich theory, generalizing results on words.
- High level model of synchronization.
- Simple: no buffers (Turing Powerful).

Deterministic Zielonka (Asynchronous) Automata



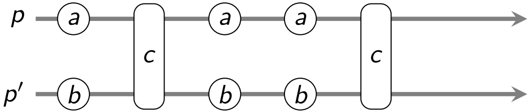
$\mathcal{A} = \langle \{S_p\}_{p \in \mathbb{P}}, s_{in}, \{\delta_a\}_{a \in \Sigma} \rangle$

- Finite set of processes: $\mathbb{P} = \{p, p'\}$.
- Distributed alphabet: Σ finite, $dom : a \mapsto \{p\},\ c \mapsto \{p, p'\}$
- Finite set of (local) states: $S_p = \{s_0, s_1, s_2\}, S_{p'} = \{s'_0, s'_1, s'_2\}$.
- $s_{in} = (s_0, s'_0)$.
- Deterministic transition functions:

$$\begin{aligned} \delta_c : S_p \times S_{p'} &\to S_p \times S_{p'} \\ (s_1, s'_1) &\mapsto (s_0, s'_0) \\ (s_2, s'_2) &\mapsto (s_0, s'_0) \end{aligned}$$

$$L(\mathcal{A}) = ((shuffle(ab) + shuffle(aabb)) \cdot c)^* \cdots$$

Traces



- Order of execution of a, b irrelevant. Independence relation I.
- $(a, b) \in I$ iff $dom(a) \cap dom(b) = \emptyset$. Induces congruence $\sim_I$ on Σ^*.
- Trace: an equivalence class $[w]_I$. $L(\mathcal{A})$ is trace-closed.

Zielonka's Theorem ('87)

Every regular trace-closed language can be recognized by a deterministic Zielonka automaton (of exponential size in $\mathbb{P}$, Genest et al. '09).

Control Problem

Variant of the control problem in [Ramadge and Wonham, '89].

Given

- Distributed alphabet, but partitioned: $\Sigma = \Sigma^{sys} \cup \Sigma^{env}$
- Deterministic Zielonka automaton P (the plant)

Find

- Another Zielonka automaton C over the same alphabet (i.e. same processes) such that the product $P \times C$ satisfies a given specification.
- C cannot block environment actions.

Assuming

- All environment actions are local.
- All synchronization actions are binary: $\forall a \in \Sigma, |dom(a)| \leq 2$.

Game Formulation

Finding a controller equivalent to finding a distributed strategy in a game between system and environment.

The Game

- Arena: Zielonka automaton $\mathcal{A}$.
- At each step: each process proposes a set of actions, environment chooses which to execute.
- A play is a trace from $L(\mathcal{A})$ (finite/infinite).
- Strategy: $\sigma = (\sigma_p)_{p \in \mathbb{P}}$, where $\sigma_p : Plays_p(\mathcal{A}) \to 2^{\Sigma_p^{sys}}$.
- The set of σ-plays is defined to contain ϵ, and for all σ-plays t:
  - if $a \in \Sigma^{env}$, $ta \in L(\mathcal{A})$ then ta is a σ-play;
  - if $a \in \Sigma^{sys}$, $ta \in L(\mathcal{A})$ and $a \in \sigma_p(t)$ for all $p \in dom(a)$ then ta is a σ-play.

Problem Statement

Winning Condition

- Stated on maximal plays.
- Local reachability condition: Each process has $F_p \subseteq S_P$. We assume that final states are absorbing, i.e. if $s_p \xrightarrow{a} s'_p$ and $s_p \in F_p$ then $s'_p \in F_p$.

Equivalence

It is easy to see that a controller defines a strategy. Conversely, a winning strategy can be transformed to a correct controller (hard).

Distributed Control Problem

Given a plant (Zielonka automaton) $\mathcal{A}$, and a local reachability condition $(F_p)_{p \in \mathbb{P}}$ determine if there is a distributed strategy $\sigma = (\sigma_p)_{p \in \mathbb{P}}$ such that every maximal σ-play t eventually ends in F_p for all p.

Results

Decidability is open in the general case.

- Previous Results:
  - [Madhusudhan & Thiagarajan 2002] Decidable with strong restriction on local strategies.
  - [Gastin & Lerman & Zeitoun 2004] Decidable with restriction on alphabet of actions (co-graphs).
  - [Madhusudhan & Thiagarajan 2005] Decidable with restriction on plant.

Related Work

Define communication graph on $\mathbb{P}$ with edge $\{p, q\}$ if there exists $a \in \Sigma$ with $dom(a) = \{p, q\}$.

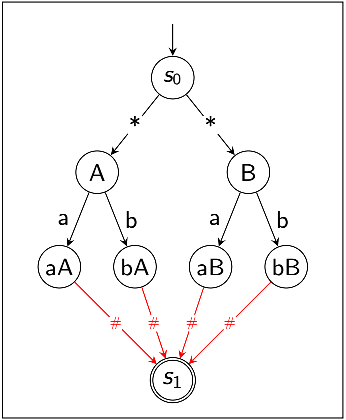
- [Genest & Gimbert & Muscholl & Walukiewicz] If the communication graph is acyclic, the control problem is decidable (non-elementary complexity). Assumes final states are blocking.

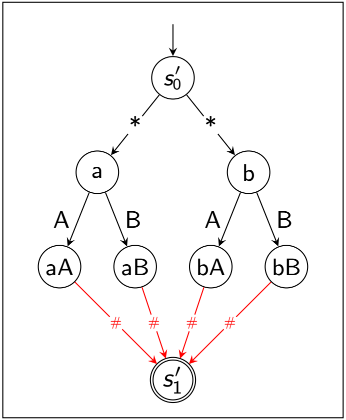
In This Talk:

Special case:

- Communication graph a tree of depth 1.
- Real world: client-server architecture.
- EXPTIME complexity algorithm to decide and recover strategy.
- Construction generalizes to other winning conditions (non-blocking, Büchi).

Example

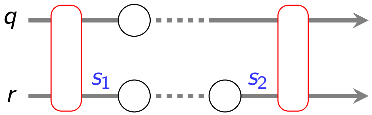


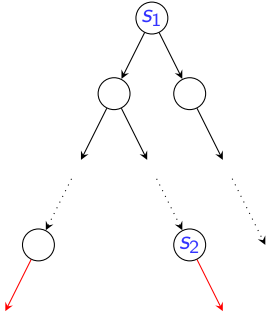


$$\delta_{\#} : (xX, yY) \mapsto (s_1, s_1') \Leftrightarrow x = y \text{ or } X = Y.$$

Construction/Reduction/Proof

- Basic idea: reduce distributed game $\mathcal{A} \to$ sequential reachability game $\mathcal{A}'$.
- Root process q "simulates" child processes $r_1, \ldots, r_k$.
- Key lemma: can simulate actions of a child between two synchronizations by a positional strategy.





Construction of $\mathcal{A}'$

$\boxed{\text{Root}\langle s_q \ , (s_j, \tau_j)_{1 \leq j \leq k} \rangle}$

- $s_q \in S_q$, $s_i \in S_{r_i}$, τ_i local positional strategy for r_i

Construction of $\mathcal{A}'$

$$\mathrm{Root}\langle s_q\ , (s_j, \tau_j)_{1\leq j\leq k}\rangle \xrightarrow[\forall B \subseteq \Sigma_q^{sys}]{\texttt{choice}(B)} \mathrm{Env}\langle s_q, B, (s_j, \tau_j)_{1\leq j\leq k}\rangle$$

- $s_q \in S_q$, $s_i \in S_{r_i}$, τ_i local positional strategy for r_i

Construction of $\mathcal{A}'$

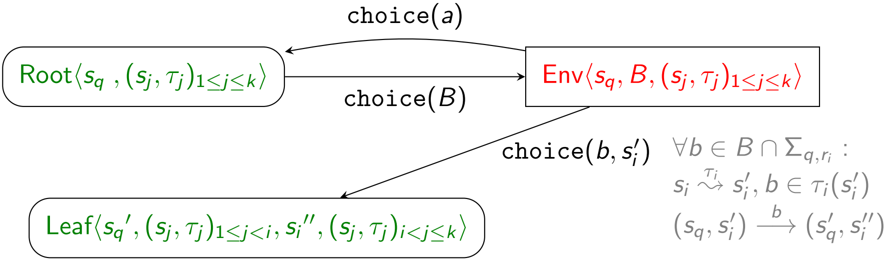
$$\forall a \in (B \cap \Sigma_q^{loc}) \cup \Sigma_q^{env} : s_q \xrightarrow{a} s_q'$$

$$\begin{tikzcd}
\mathrm{Root}\langle {s_q}', (s_j, \tau_j)_{1 \le j \le k} \rangle \arrow[rr, bend left=0, shift left, "\texttt{choice}(B)"'] & & \mathrm{Env}\langle s_q, B, (s_j, \tau_j)_{1 \le j \le k} \rangle \arrow[ll, bend right, "\texttt{choice}(a)"']
\end{tikzcd}$$

Environment either chooses a local action a for q

- $s_q \in S_q$, $s_i \in S_{r_i}$, τ_i local positional strategy for r_i

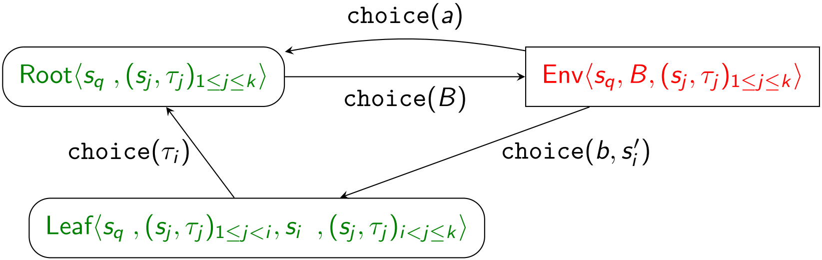
Construction of $\mathcal{A}'$



Or chooses a synchronization action b between q and r_i.

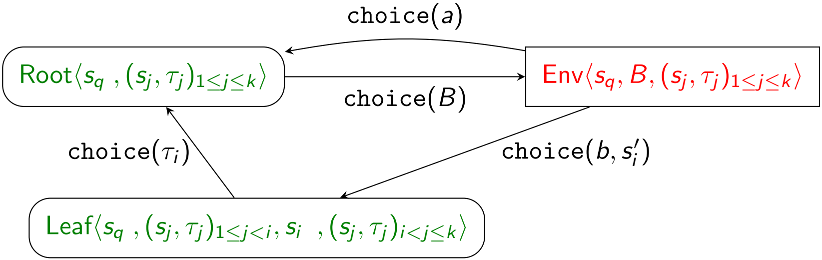
- $s_q \in S_q$, $s_i \in S_{r_i}$, τ_i local positional strategy for r_i

Construction of $\mathcal{A}'$



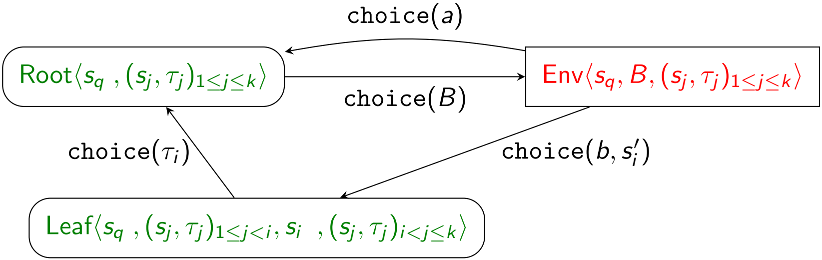
- $s_q \in S_q$, $s_i \in S_{r_i}$, τ_i local positional strategy for r_i

Construction of $\mathcal{A}'$



- $s_q \in S_q$, $s_i \in S_{r_i}$, τ_i local positional strategy for r_i
- Good τ_i : every infinite local play from s_i ultimately in F_{r_i}.

Construction of $\mathcal{A}'$



- $s_q \in S_q$, $s_i \in S_{r_i}$, τ_i local positional strategy for r_i
- Good τ_i : every infinite local play from s_i ultimately in F_{r_i}.
- Final states: $s_q \in F_q$, all local maximal finite τ_i-plays from s_i end in F_{r_i}.

Proof

Theorem

The system has a winning strategy for $\mathcal{A}, (F_p)_{p \in \mathbb{P}}$ if and only if the system has a winning strategy for $\mathcal{A}', F$.

Recovering Strategy

- q plays the actions as given by $\texttt{choice}(a), \texttt{choice}(a, s_i')$, etc
- Each r_i plays according to τ_i.

Implementation Demo

- Implemented as an extension to a Java program called GAVS+, by Chih-Hong Cheng, TU München.

Conclusion

- Generalize to other winning conditions: Büchi. Ongoing, have preliminary results.
- Other research directions.

Thank you :)