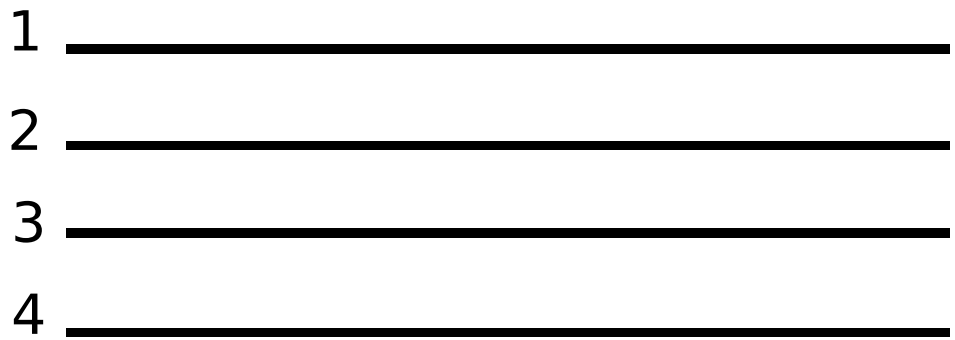
A Class of Zielonka Automata with a Decidable Controller Synthesis Problem

Hugo Gimbert, CNRS, LaBRI, Université de Bordeaux

September 7, 2016

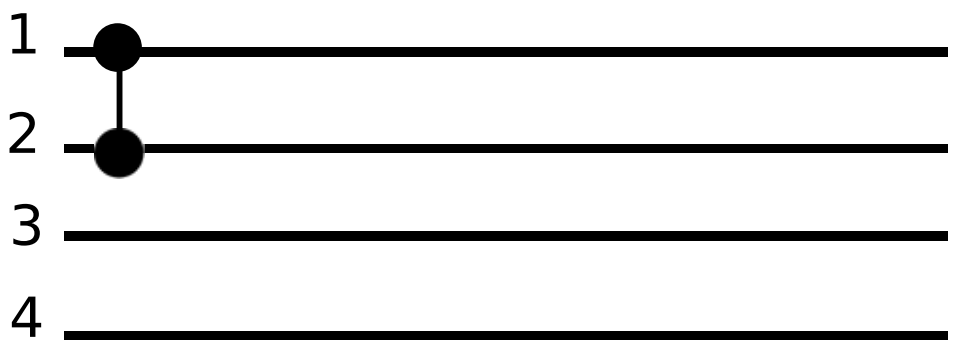
The thread-free language

Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



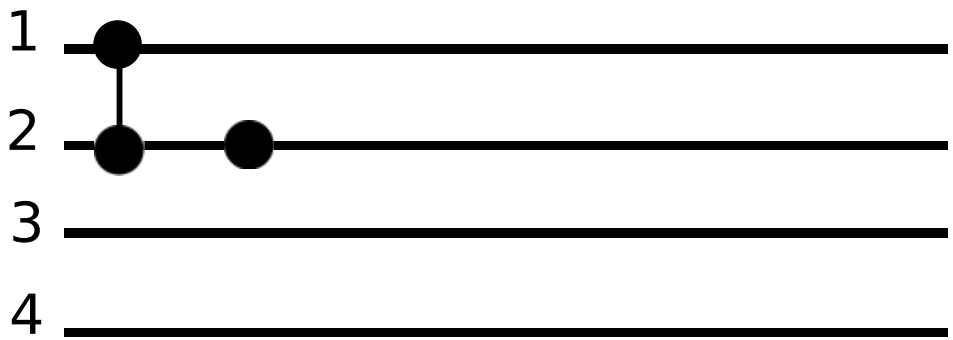
The thread-free language

Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



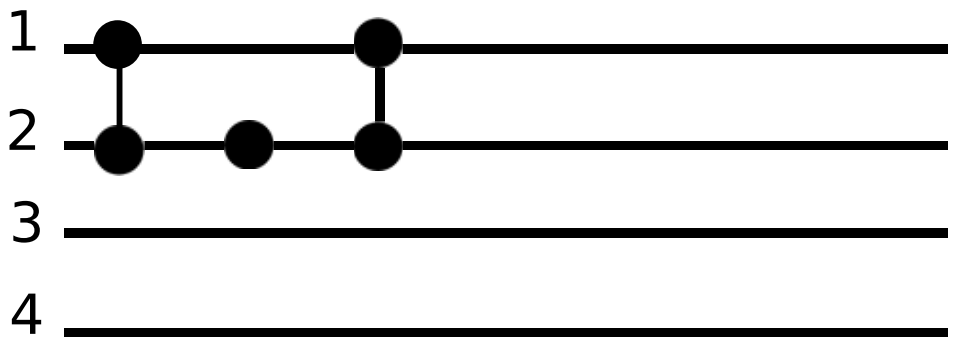
The thread-free language

Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



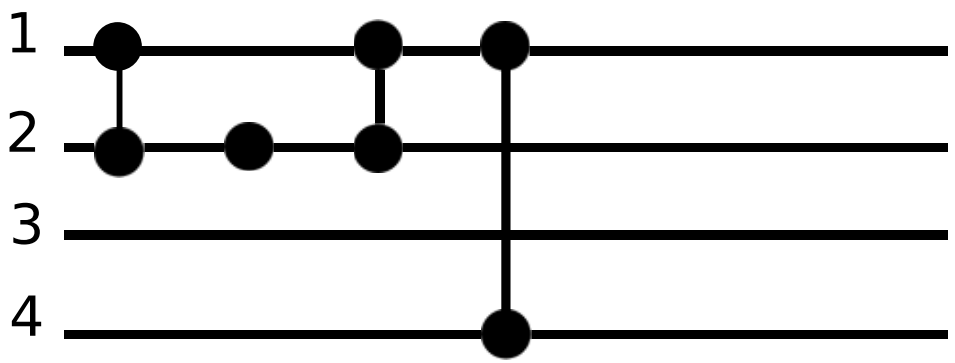
The thread-free language

Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



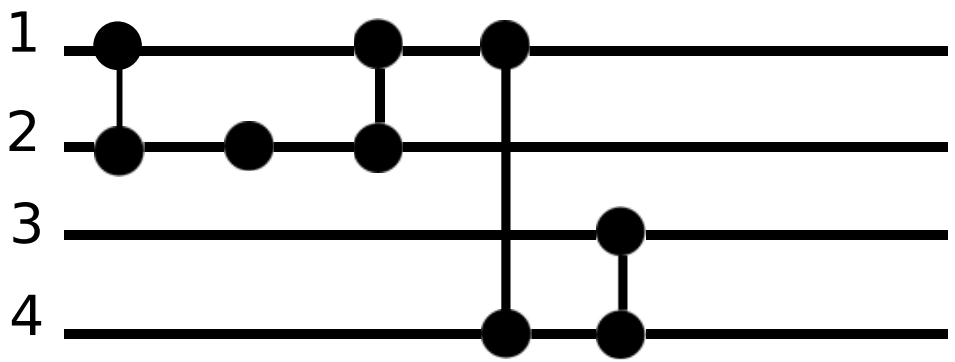
The thread-free language

Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



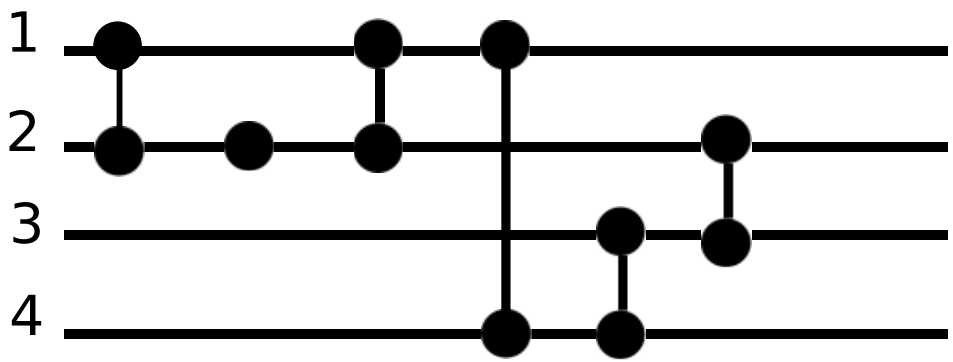
The thread-free language

Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



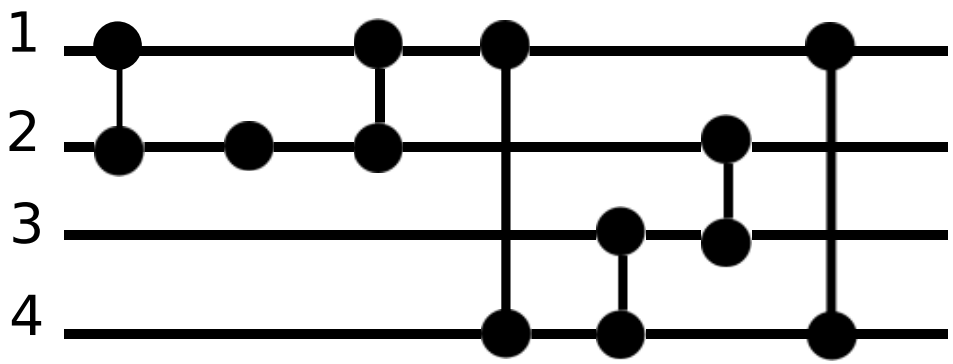
The thread-free language

Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



The thread-free language

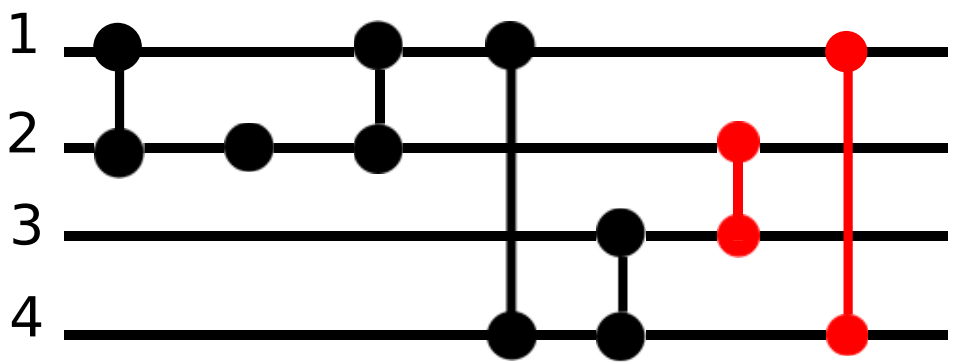
Set of processes $\mathbb{P} = \{1, 2, 3, 4\}$
Alphabet $A = \{a_{p,q} \mid p, q \in \mathbb{P}\}$



The thread-free language

Language: two consecutive letters should touch each other.

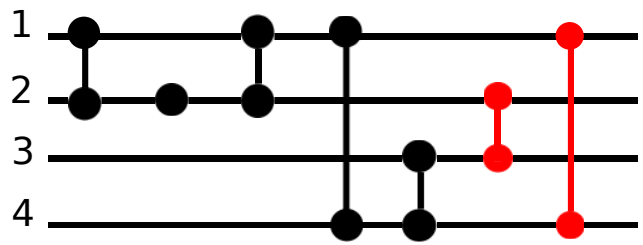
$$L = \{a_{p_0,q_0} a_{p_1,q_1} \cdots a_{p_n,q_n} \mid \forall i, \{p_i, q_i\} \cap \{p_{i+1}, q_{i+1}\} \neq \emptyset\}$$



Interpretation: $a_{p,q}$ = synchro of processes p and q.
Forbids parallel events.

Zielonka automaton

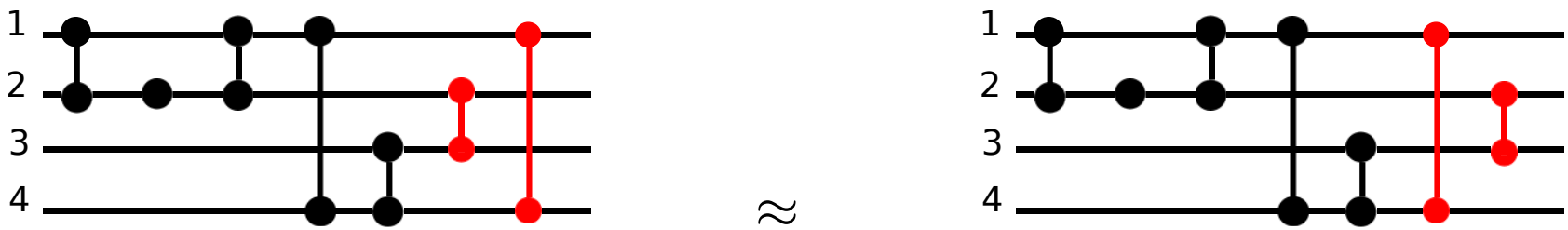
Recognize L in a distributed way?
Zielonka automaton.
Processes $=$ finite automata $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$.
Letter a has a domain $\mathrm{dom}(a) \subseteq \mathbb{P}$



$a_{1,2}$ synchronizes $\mathcal{A}_1$ and $\mathcal{A}_2$ on transition $(q_1, q_2, a_{1,2}, q_1', q_2')$
Asynchronous. Deterministic. Local final states.

Commutation

Closure by commutation.



Zielonka automaton: rational language of Mazurkiewicz traces.

Asynchronous distributed games

Supervisory control theory for discrete event systems:
find a permissive controller which guarantees a correct behaviour of the plant (Ramadge Wonham).

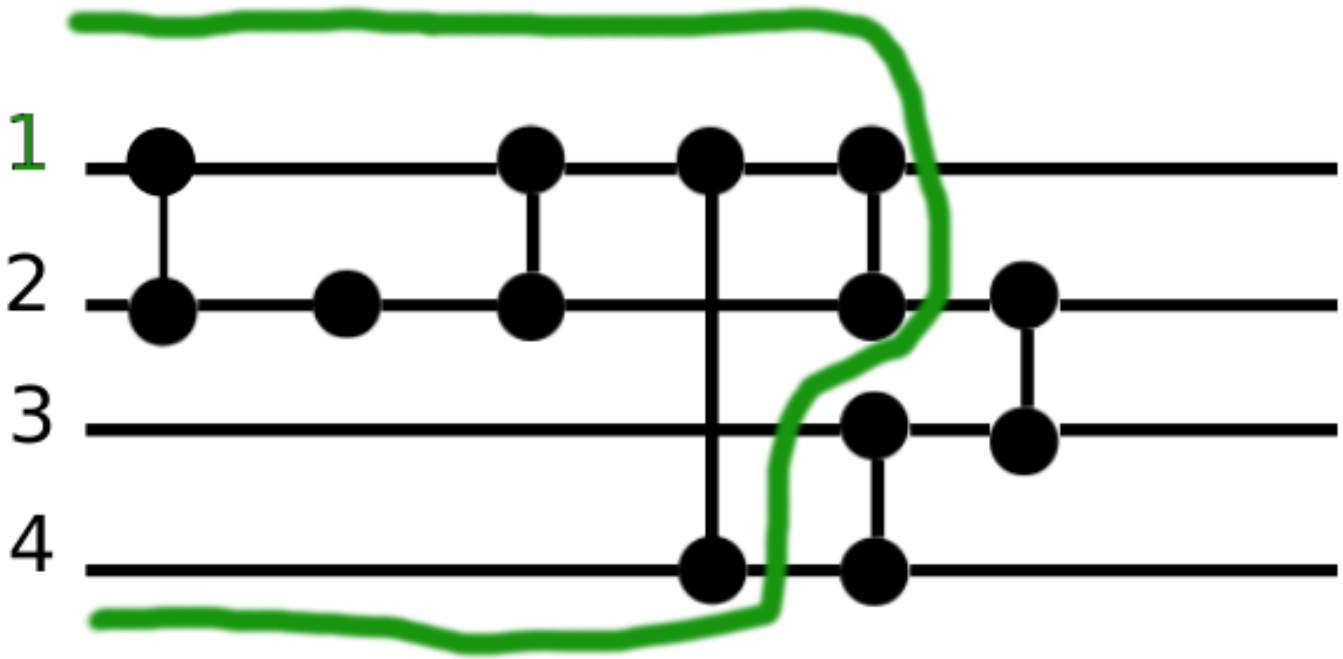
Plant: Zielonka automaton = Distributed game

Controller: distributed strategy. Each process chooses letters = actions depending on its view of the play.

Environment actions are uncontrollable. $A = A_c \sqcup A_e$. Processes cannot prevent environnment actions to be played.

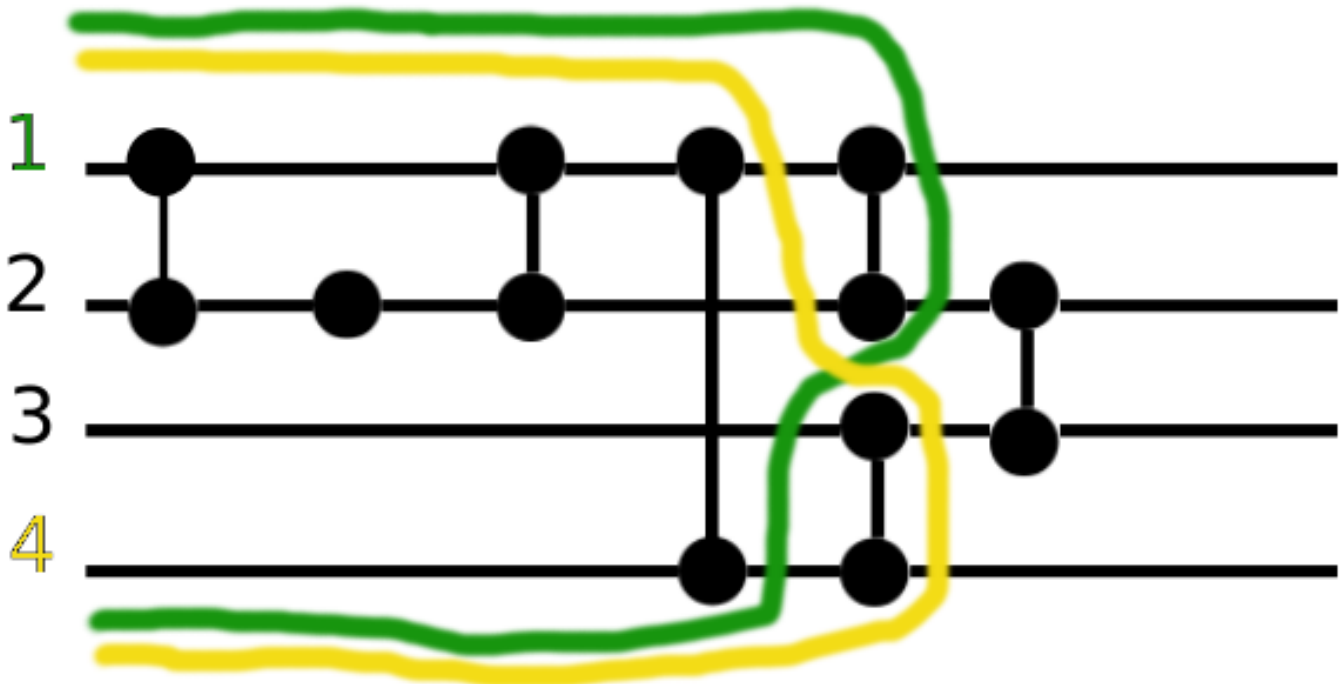
Specification: Every process p terminates in a final state.

Views and strategies



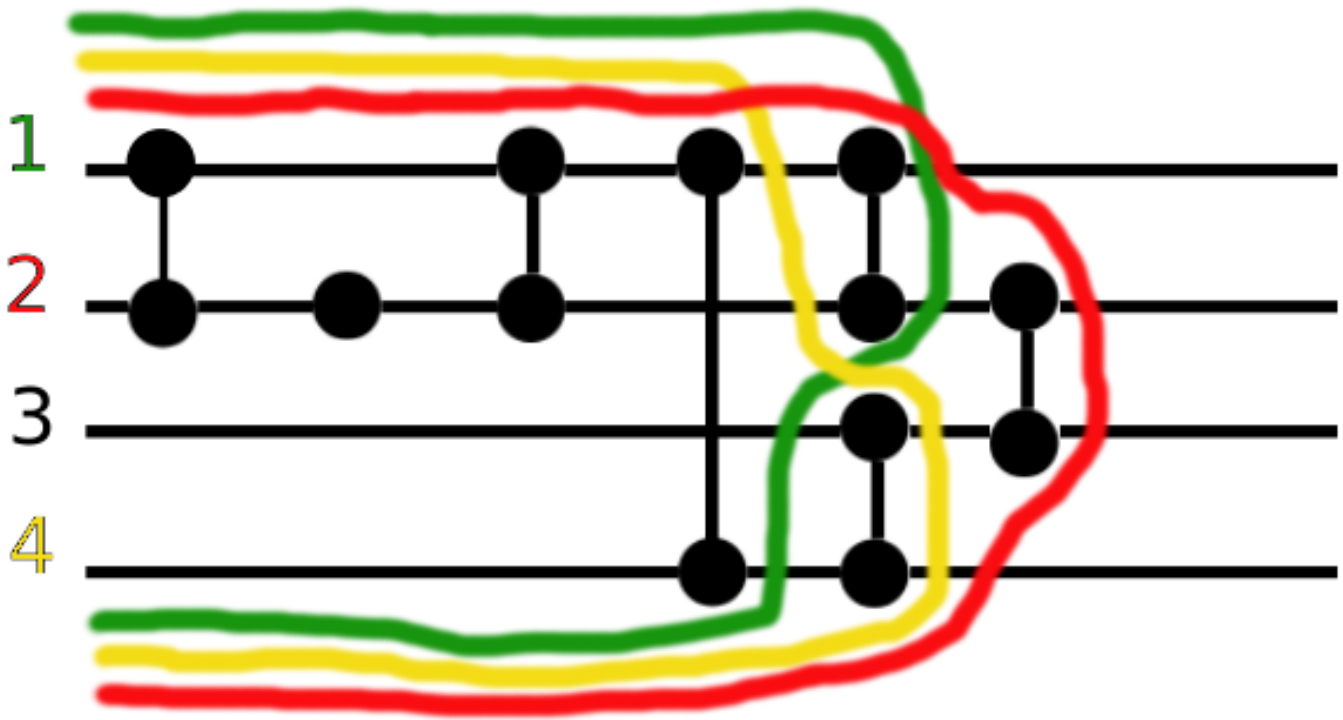
View $\partial_1(u)$ of process 1

Views and strategies



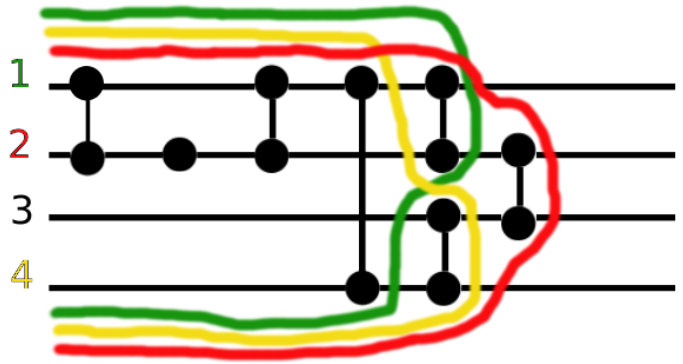
View $\partial_1(u)$ of process 1
View $\partial_4(u)$ of process 4

Views and strategies



View $\partial_1(u)$ of process 1
View $\partial_4(u)$ of process 4
View $\partial_2(u)$ of process 2

Views and strategies



Strategy σ_p of process p enables a set of actions $\sigma_p(u) \subseteq A$.

- depends on the view

$$\sigma_p(u) = \sigma_p(\partial_p(u)) \ .$$

- allows uncontrollable actions

$$A_e \subseteq \sigma(u).$$

Play: action a may be played if if every $p \in \mathrm{dom}(a)$ plays a.
Non-determinism. One strategy, many plays.
No choice = end of the game.

A decision problem

Winning strategy = ensures end of the game in final states.

Controller synthesis Decide the existence of a winning strategy. (And compute it).

Decidability = open problem

Non-trivial because:

- the set of strategies $\sigma : A^* \to 2^A$ is infinite.
- no finite memory result.

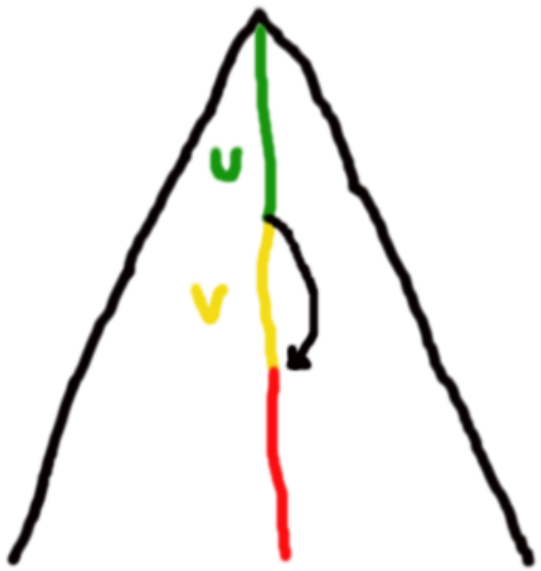
Decidable cases

- Connectedly communicating plants (MSO-specifications)
  [Madhusudan, Thiagarajan, Yang, 05]
  For some k, if process p plays for k consecutive steps without having news about q then p will never hear about q anymore.
- Series-parallel alphabets (action-based games)
  [Gastin, Lerman, Zeitoun 04]
  The dependency graph of actions is series-parallel
- Acyclic games [Genest,Gimbert,Muscholl,Walukiewicz 13]
  The dependency graph of processes is acyclic.
- Broadcast games [Gimbert 16]
  Inductive decomposition into subgames such that every play long enough in a subgame contains a broadcast.

Broadcast games = proof technique encompass all known decidable classes.

Taking shortcuts

Cut and paste in a strategy tree.



$$\sigma_{u,v}(x) = \begin{cases} \sigma(x) & \text{if } u \not\sqsubseteq x \\ \sigma(xuv) & \text{if } x = uy \end{cases}$$

Assume

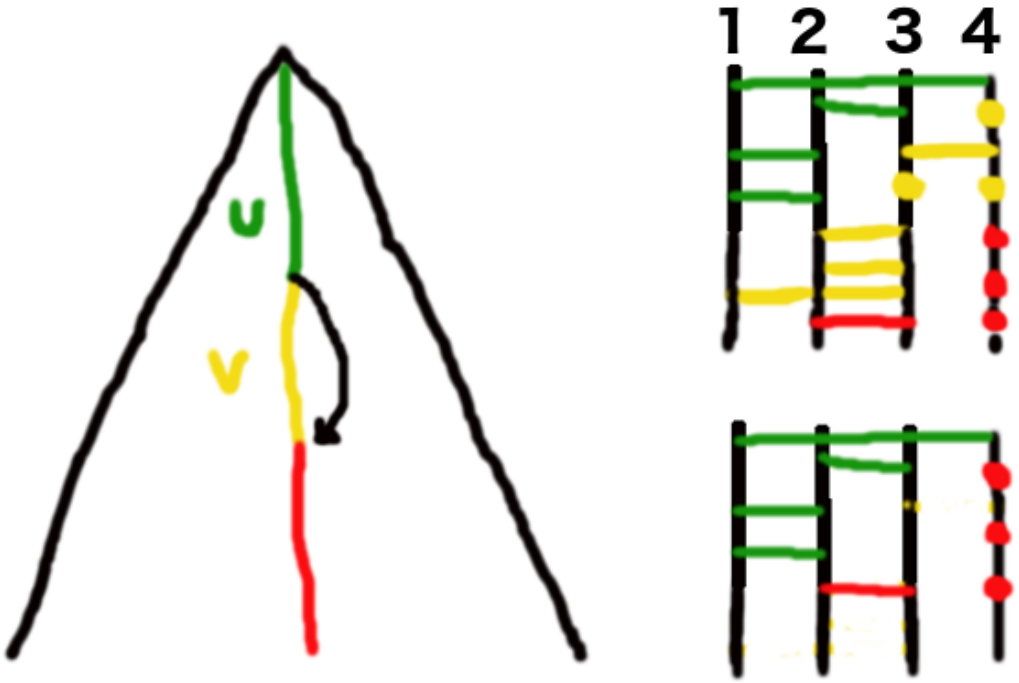
- one process
- same state in u and uv.

Taking shortcuts again and again leads to positional strategy.

Taking shortcuts

In general there is no reason for a shortcut to be a strategy maybe

$$\sigma'_p(ux) \neq \sigma'_p(\partial_p(ux))$$



Consistent shortcut

Definition

Action b and processes $\mathbb{Q} \subseteq \mathbb{P}$. Play ub is a $\mathbb{Q}$-broadcast if no process of $\mathbb{Q}$ can learn about ub from a process from $\mathbb{P} \setminus \mathbb{Q}$.

Broadcasts allow sychronization of processes in $\mathbb{Q}$.

Definition

Consistent shortcut $= uv \in \sigma$ and $\mathbb{Q} \subseteq \mathbb{P}$

only processes of $\mathbb{Q}$ play in v

u and uv are $\mathbb{Q}$-broadcasts,

processes in $\mathbb{Q}$ have the same control states in u and uv,

processes in $\mathbb{Q}$ play the same way in parallel of u and uv .

Existence of useless threads

Lemma

If (u, v) is a consistent shortcut then $\sigma_{u,v}$ is also a strategy.
If σ is winning then $\sigma_{x,y}$ is winning as well.
And $\sigma_{x,y}$ has shorter duration.

Decidability: in a game where broadcasts occur periodically in subgames, we can infer a bound on the size of a winning strategy.

Conclusion

Broadcast games: general class of distributed asynchronous games with a decidable synthesis problem.

Encompasses known decidable examples

General case open

Ring game with 5 players or synchronizing games?