Distribution of Reconfiguration Languages maintaining Tree-like Communication Topology

Daniel Hausmann, Mathieu Lehaut, Nir Piterman

University of Gothenburg

ATVA 2024

Synthesis for distributed systems

Synthesis Problem

Input: A specification φ
Output: A program P satisfying φ

Synthesis for distributed systems

Synthesis Problem

Input: A specification φ
Output: A program P satisfying φ

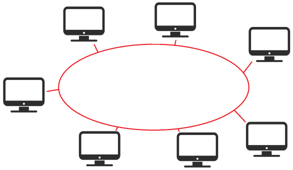
- Great if possible, but very hard.

Synthesis for distributed systems

Synthesis Problem

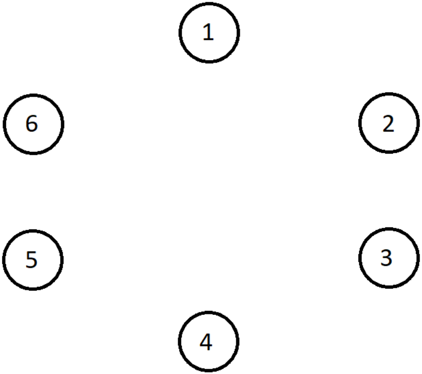
Input: A specification φ
Output: A program P satisfying φ

- Great if possible, but very hard.
- Distributed systems makes it even harder!
- Specifications are centralized, programs are distributed.



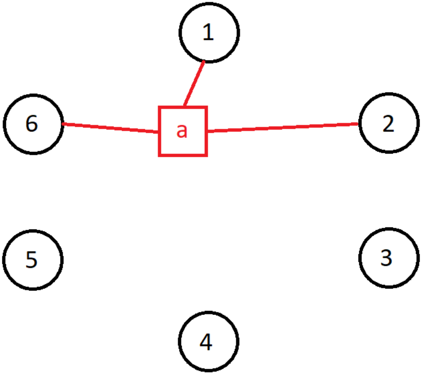
Distributed setting

Independent processes



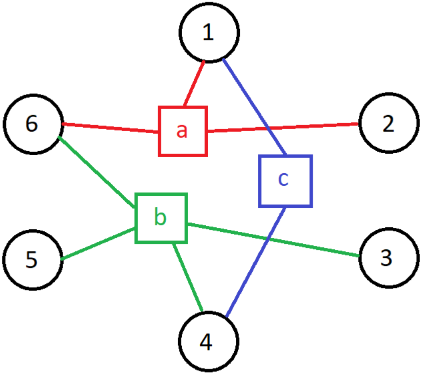
Distributed setting

Independent processes communicating over channels



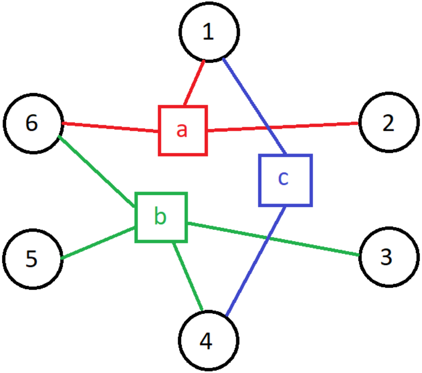
Distributed setting

Independent processes communicating over channels



Distributed setting

Independent processes communicating over channels



Some assumptions

- Asynchronous system
- Rendez-vous communication
- No sender/receiver distinction

Automaton model for distributed systems

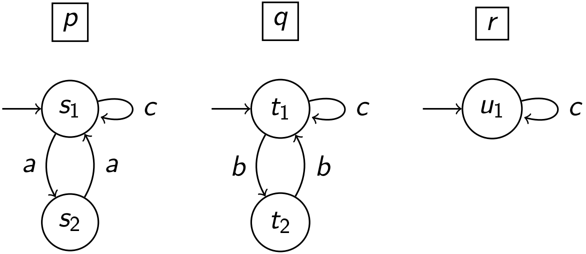
Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

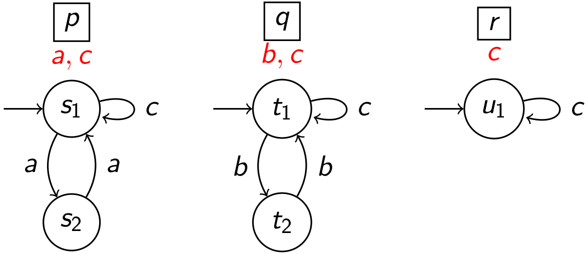
Fix set of processes, channels, and communication topology



Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

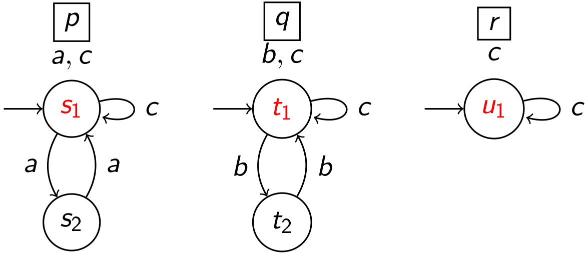
Fix set of processes, channels, and communication topology



Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology

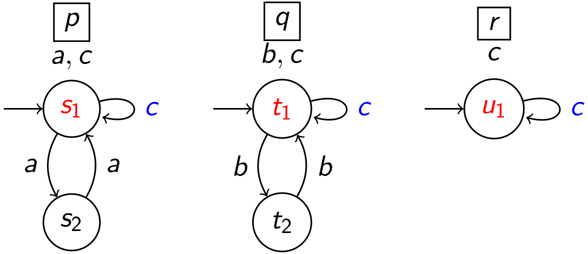


$\rho = (s_1, t_1, u_1)$

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology

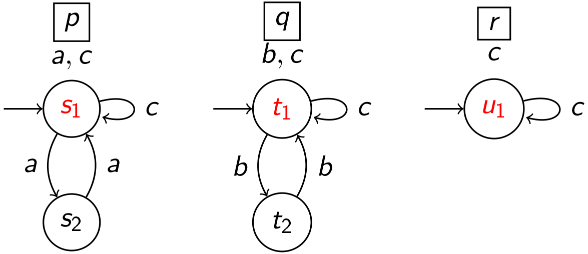


$\rho = (s_1, t_1, u_1) \to^c$

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology

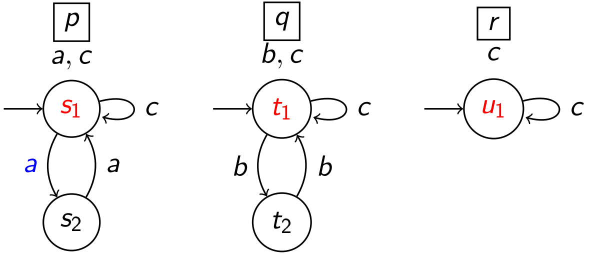


$\rho = (s_1, t_1, u_1) \to^c (s_1, t_1, u_1)$

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology

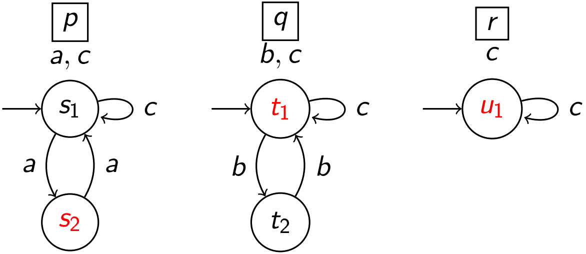


$\rho = (s_1, t_1, u_1) \rightarrow^{c} (s_1, t_1, u_1) \rightarrow^{a}$

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology

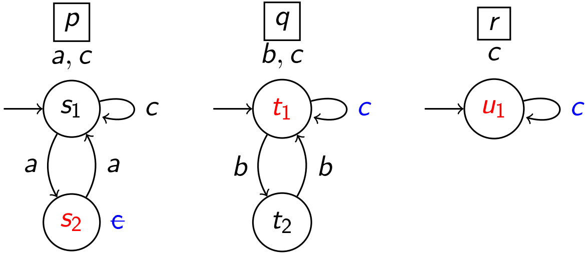


$$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1) \to^{a} (s_2, t_1, u_1)$$

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology

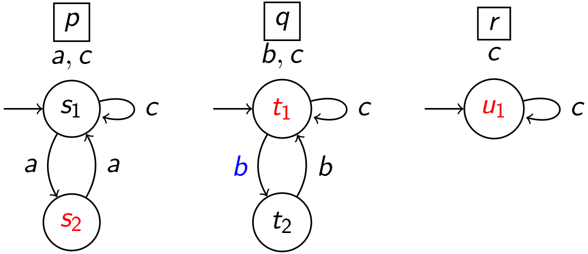


$$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1) \to^{a} (s_2, t_1, u_1) \not\to^{c}$$

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology



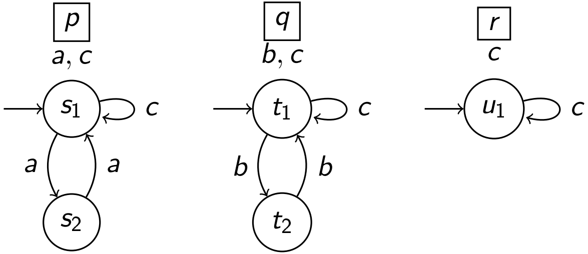
$\rho = (s_1, t_1, u_1) \rightarrow^{c} (s_1, t_1, u_1) \rightarrow^{a} (s_2, t_1, u_1) \rightarrow^{b} \ldots$

$w = cab\ldots$

Automaton model for distributed systems

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication topology



$\rho = (s_1, t_1, u_1) \rightarrow^c (s_1, t_1, u_1) \rightarrow^a (s_2, t_1, u_1) \rightarrow^b \ldots$

$w = cab\ldots$ Language: all c preceded by even number of a & b

Zielonka's distributivity theorem

Theorem [Zielonka, 1987]

Given a communication topology and a diamond-closed DFA $\mathcal{A}$, one can build an asynchronous automaton $\mathcal{A}_{||}$ recognizing the same language as $\mathcal{A}$.

Zielonka's distributivity theorem

Theorem [Zielonka, 1987]

Given a communication topology and a diamond-closed DFA $\mathcal{A}$, one can build an asynchronous automaton $\mathcal{A}_{||}$ recognizing the same language as $\mathcal{A}$.

- Complexity: exp. in $|Proc|$, poly. in $|\mathcal{A}|$ [Genest et al., 2010]

Zielonka's distributivity theorem

Theorem [Zielonka, 1987]

Given a communication topology and a diamond-closed DFA $\mathcal{A}$, one can build an asynchronous automaton $\mathcal{A}_{||}$ recognizing the same language as $\mathcal{A}$.

- Complexity: exp. in $|Proc|$, poly. in $|\mathcal{A}|$ [Genest et al., 2010]
- On trees: $O(|\mathcal{A}|^2)$ construction [Krishna & Muscholl, 2013]
(tree = binary channels and acyclic topology)

Reconfiguration

▶ What if processes could change dynamically the channels they listen to? Applications in:

- Swarm protocols (connected based on distance)
- Privacy (need-to-know)
- Energy constraints (turn off communications if not needed)

Reconfiguration

▶ What if processes could change dynamically the channels they listen to? Applications in:

- Swarm protocols (connected based on distance)
- Privacy (need-to-know)
- Energy constraints (turn off communications if not needed)

Goal of this presentation

▶ Adapt the tree construction to this setting:

- Input language contains instructions for reconfiguration
- Output automaton should implement them only with local information

Reconfigurable automaton model

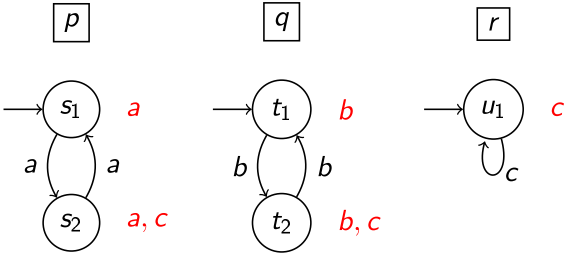
Reconfigurable Asynchronous Automata (RAA)

Fix set of processes, channels, and ~~communication topology~~

Reconfigurable automaton model

Reconfigurable Asynchronous Automata (RAA)

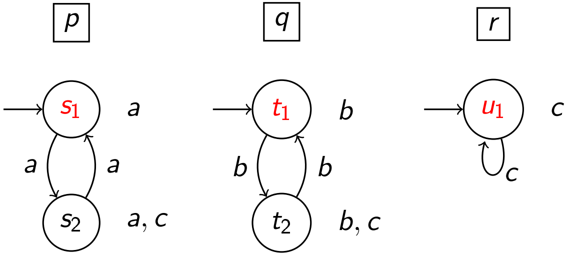
Fix set of processes, channels, and ~~communication topology~~
Communication topology is state-dependent



Reconfigurable automaton model

Reconfigurable Asynchronous Automata (RAA)

Fix set of processes, channels, and ~~communication topology~~
Communication topology is state-dependent

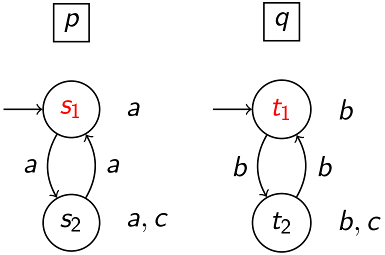


$\rho = (s_1, t_1, u_1)$

Reconfigurable automaton model

Reconfigurable Asynchronous Automata (RAA)

Fix set of processes, channels, and ~~communication topology~~
Communication topology is state-dependent

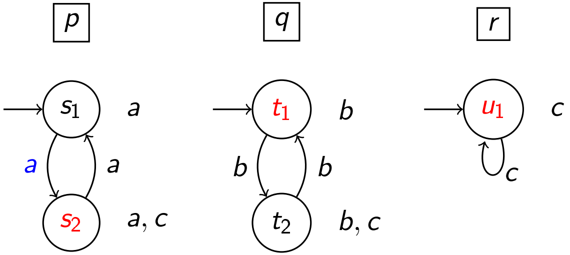


$\rho = (s_1, t_1, u_1) \rightarrow^{c} (s_1, t_1, u_1)$

Reconfigurable automaton model

Reconfigurable Asynchronous Automata (RAA)

Fix set of processes, channels, and ~~communication topology~~
Communication topology is state-dependent

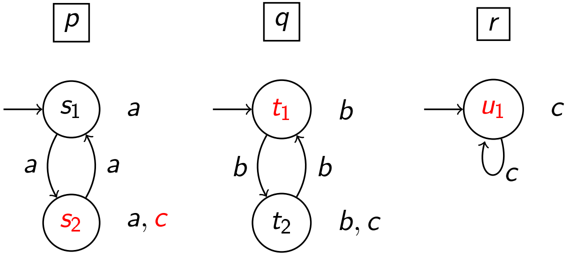


$$\rho = (s_1, t_1, u_1) \rightarrow^{c} (s_1, t_1, u_1) \rightarrow^{a} (s_2, t_1, u_1)$$

Reconfigurable automaton model

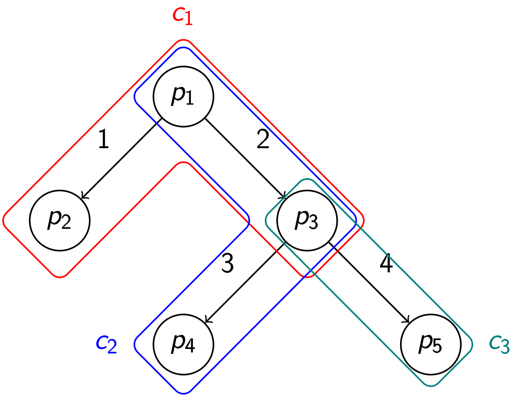
Reconfigurable Asynchronous Automata (RAA)

Fix set of processes, channels, and ~~communication topology~~
Communication topology is state-dependent



$$\rho = (s_1, t_1, u_1) \rightarrow^{c} (s_1, t_1, u_1) \rightarrow^{a} (s_2, t_1, u_1) \not\twoheadrightarrow^{c}$$

Tree-like Communication Topologies



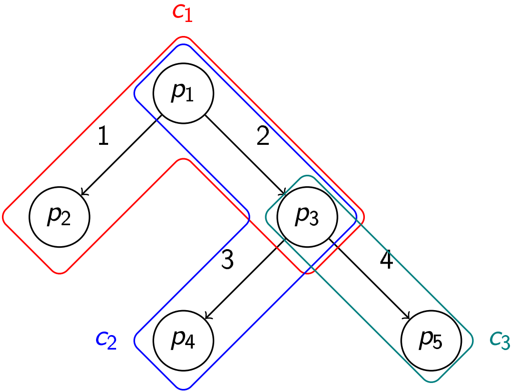
Tree-like:

- Underlying spanning tree
- Channels are connected
- Edges are covered

Reconfiguration Language

Includes reconfiguration operations with each communication:

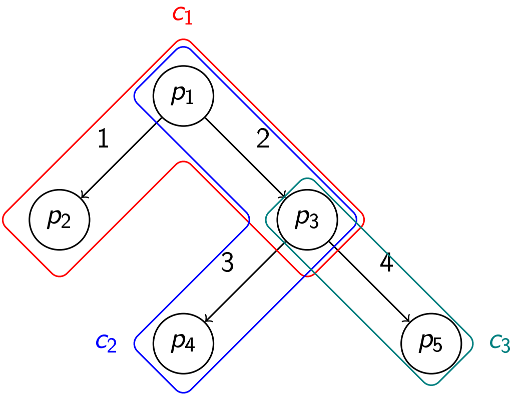
nop, swap, move, connect, disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

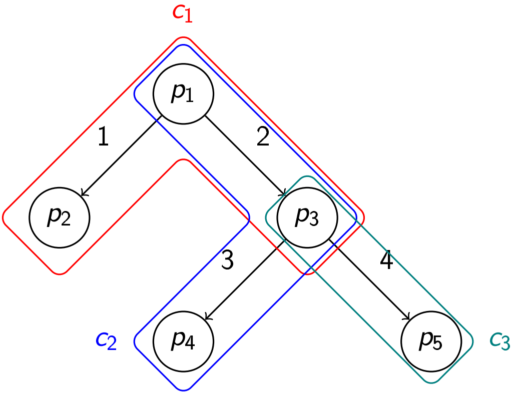
c_1:nop, swap, move, connect, disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

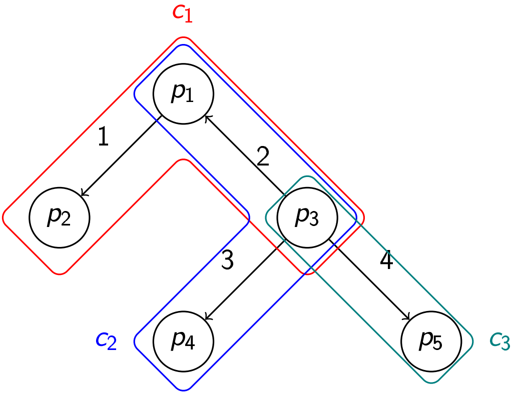
nop, c_2:swap(2), move, connect, disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

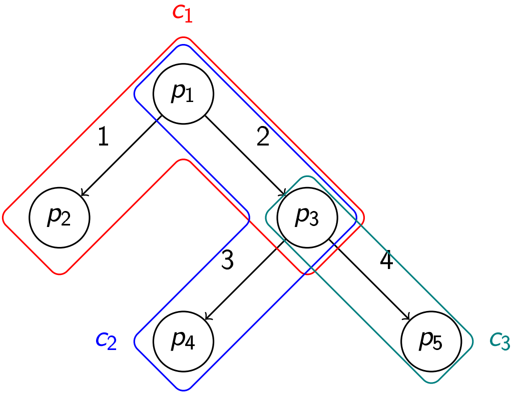
nop, c_2:swap(2), move, connect, disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

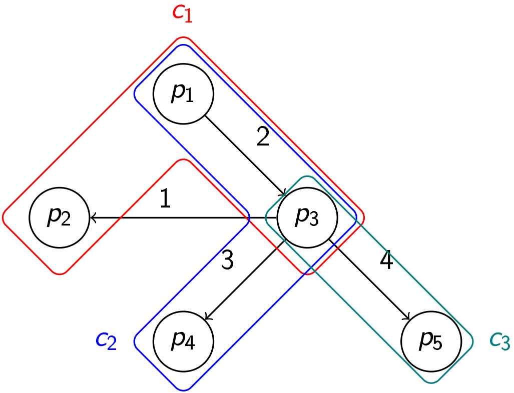
nop, swap, c_1:move$(1 \rightarrow 2)$, connect, disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

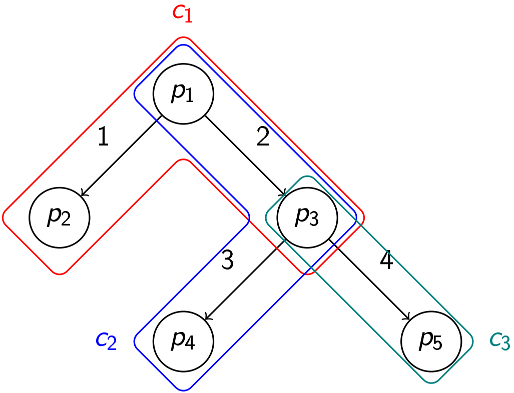
nop, swap, c_1:move$(1 \rightarrow 2)$, connect, disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

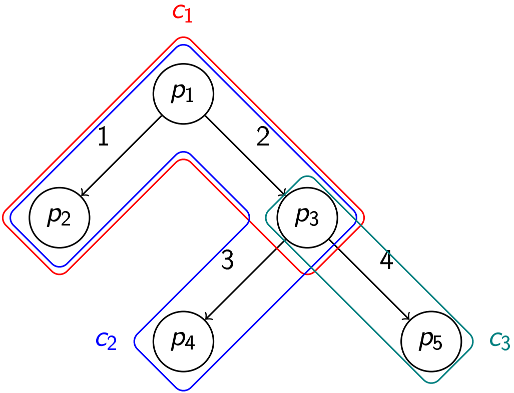
nop, swap, move, c_1:connect(1, join c_2), disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

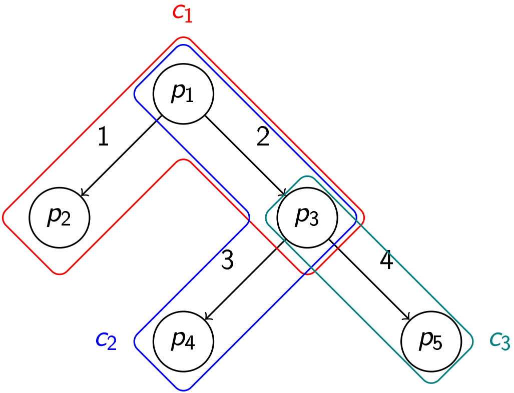
nop, swap, move, c_1:connect(1, join c_2), disconnect



Reconfiguration Language

Includes reconfiguration operations with each communication:

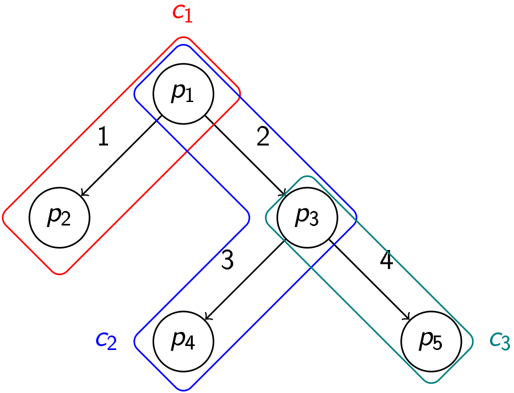
nop, swap, move, connect, c_1:disconnect(2)



Reconfiguration Language

Includes reconfiguration operations with each communication:

nop, swap, move, connect, c_1:disconnect(2)



Our result

Reminder: Zielonka's Theorem

Given a communication topology and a diamond-closed DFA $\mathcal{A}$, one can build an asynchronous automaton $\mathcal{A}_{||}$ recognizing the same language as $\mathcal{A}$.

Our result

Theorem

Given an initial tree-like topology and a diamond-closed DFA $\mathcal{A}$ for reconfiguration languages, one can build a reconfigurable asynchronous automaton $\mathcal{A}_{||}$ recognizing the same language as $\mathcal{A}$.

Our result

Theorem

Given an initial tree-like topology and a diamond-closed DFA $\mathcal{A}$ for reconfiguration languages, one can build a reconfigurable asynchronous automaton $\mathcal{A}_{||}$ recognizing the same language as $\mathcal{A}$.

- Complexity: polynomial in the size of $\mathcal{A}$

Idea of the construction

Two things to maintain: state and communication topology

Idea of the construction

Two things to maintain: state and communication topology

- State: process p maintains two states $(s_p, t_p) \in S^2$
  - s_p: most up-to-date known by p
  - t_p: state after last communication with parent of p

Idea of the construction

Two things to maintain: state and communication topology

- State: process p maintains two states $(s_p, t_p) \in S^2$
  - s_p: most up-to-date known by p
  - t_p: state after last communication with parent of p

On communication involving parent q:

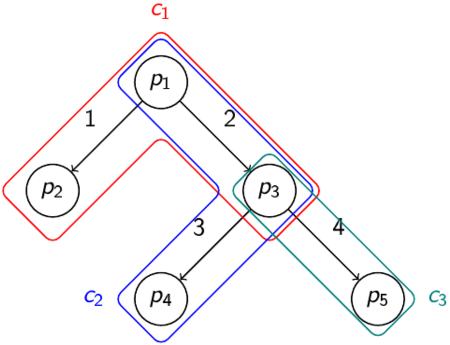
- combine s_p, s_q, and t_p to get new state s'

Need to know topology to compute new state!

Idea of the construction

Two things to maintain: state and communication topology

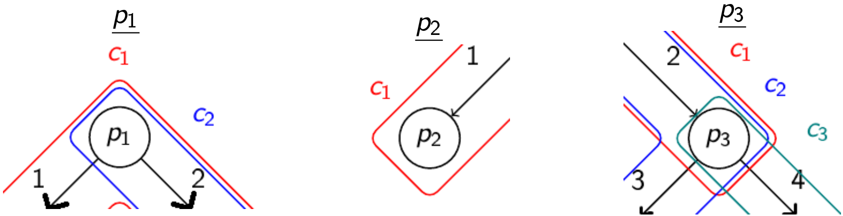
- Communication topology: maintain local view of the tree



Idea of the construction

Two things to maintain: state and communication topology

- Communication topology: maintain local view of the tree

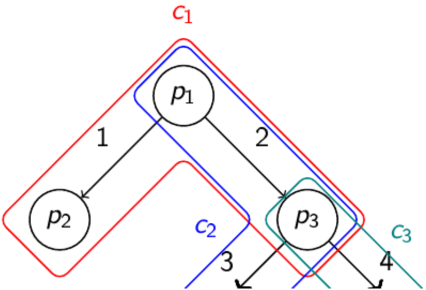


Idea of the construction

Two things to maintain: state and communication topology

- Communication topology: maintain local view of the tree

On c_1 communication, combine all local views:



Conclusion

Summary

We extend the tree topology construction for Zielonka's distributivity theorem in two directions:

- Tree topology ⇒ Tree-like topology
- Fixed topology ⇒ Reconfigurable topology

Conclusion

Summary

We extend the tree topology construction for Zielonka's distributivity theorem in two directions:

- Tree topology $\Rightarrow$ Tree-like topology
- Fixed topology $\Rightarrow$ Reconfigurable topology

Future works

- Explore more general communication topologies
- Games over reconfigurable automata

Conclusion

Summary

We extend the tree topology construction for Zielonka's distributivity theorem in two directions:

- Tree topology ⇒ Tree-like topology
- Fixed topology ⇒ Reconfigurable topology

Future works

- Explore more general communication topologies
- Games over reconfigurable automata

Thanks, questions?