Synthesis of Distributed Implementations from Centralized Specifications

Mathieu Lehaut
Joint work in progress with Daniel Hausmann and Nir Piterman

FM Retreat, 12/12/2023

What is synthesis actually?

Synthesis Problem

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

What is synthesis actually?

Synthesis Problem

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

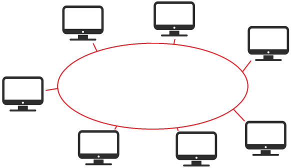
- Great if possible, but very hard.

What is synthesis actually?

Synthesis Problem

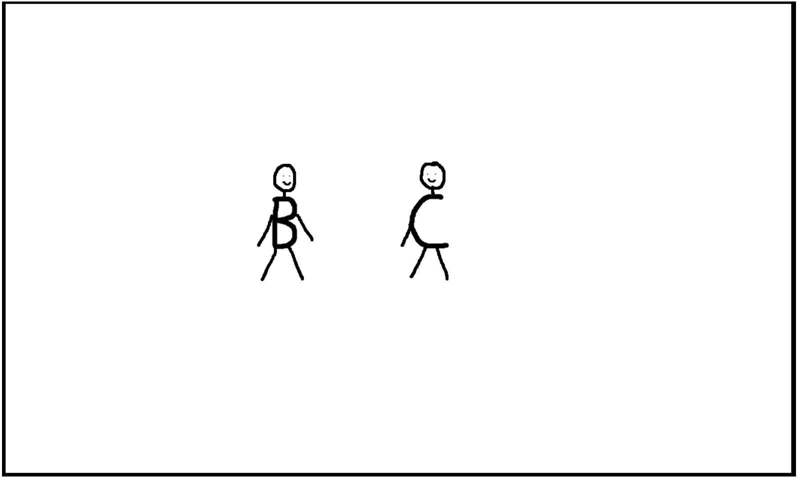
Input: A specification φ
Output: A program P satisfying φ

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

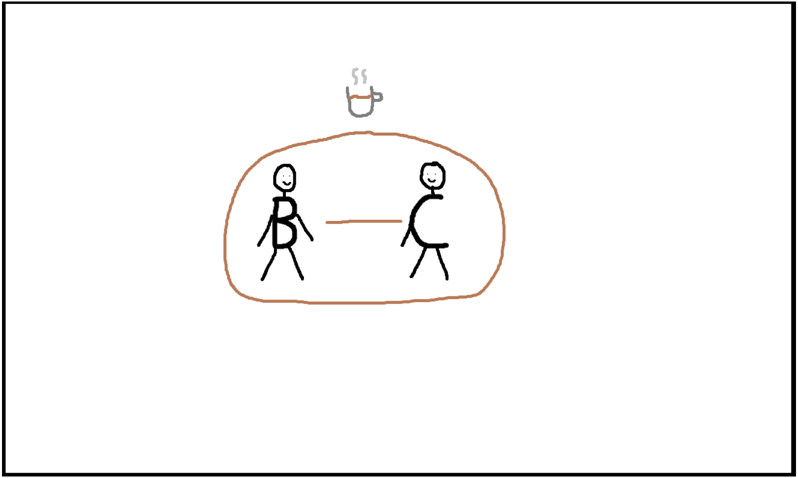


Example of a distributed protocol

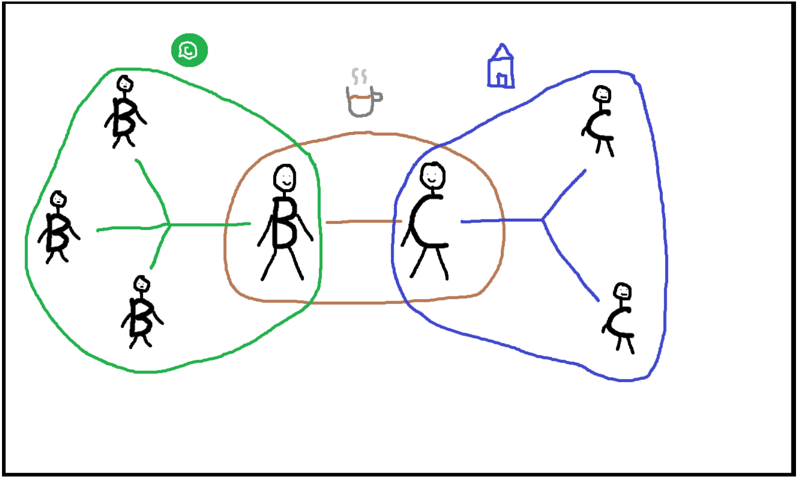
Example of a distributed protocol



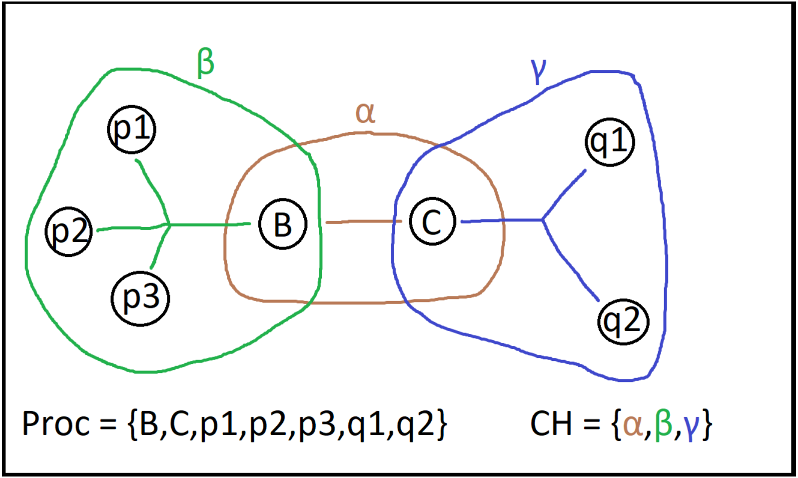
Example of a distributed protocol



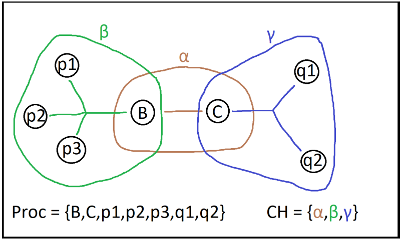
Example of a distributed protocol



Example of a distributed protocol

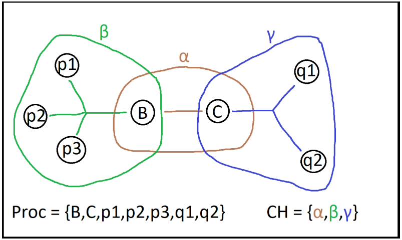


Example of a distributed protocol



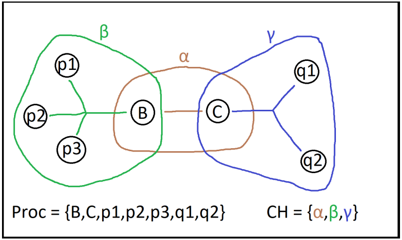
$$\mathcal{L} = \{\alpha \cdot \beta \cdot \gamma \cdot \alpha \cdot \beta \cdot \gamma, \qquad \}$$

Example of a distributed protocol



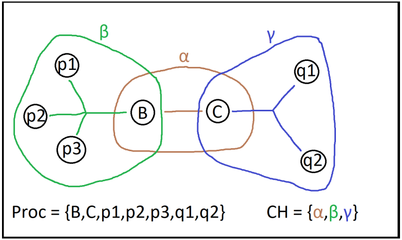
$$\mathcal{L} = \{\alpha \cdot \beta \cdot \gamma \cdot \alpha \cdot \beta \cdot \gamma, \quad \alpha \cdot \gamma \cdot \beta \cdot \alpha \cdot \beta \cdot \gamma, \quad \}$$

Example of a distributed protocol

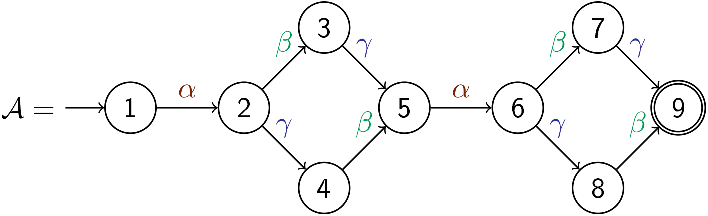


$$\mathcal{L} = \{\alpha \cdot \beta \cdot \gamma \cdot \alpha \cdot \beta \cdot \gamma, \\ \alpha \cdot \gamma \cdot \beta \cdot \alpha \cdot \beta \cdot \gamma, \\ \alpha \cdot \beta \cdot \gamma \cdot \alpha \cdot \gamma \cdot \beta, \\ \alpha \cdot \gamma \cdot \beta \cdot \alpha \cdot \gamma \cdot \beta\}$$

Example of a distributed protocol

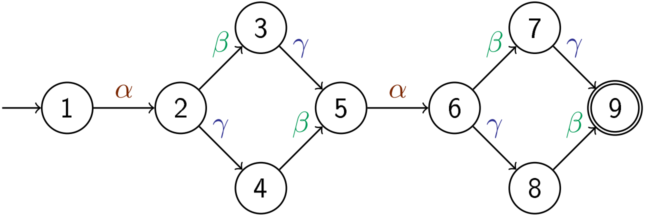


$$\mathcal{L} = \{\alpha\cdot\beta\cdot\gamma\cdot\alpha\cdot\beta\cdot\gamma,\ \alpha\cdot\gamma\cdot\beta\cdot\alpha\cdot\beta\cdot\gamma,\ \alpha\cdot\beta\cdot\gamma\cdot\alpha\cdot\gamma\cdot\beta,\ \alpha\cdot\gamma\cdot\beta\cdot\alpha\cdot\gamma\cdot\beta\}$$



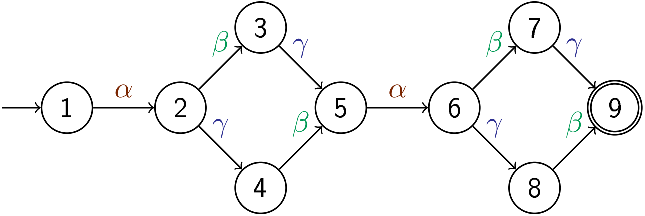
Distributed view

$\mathcal{A} =$



$$\rho = \alpha \cdot \beta \cdot \gamma \cdot \alpha \cdot \beta \cdot \gamma$$

Distributed view

$\mathcal{A} =$ 

$\rho = \varepsilon$

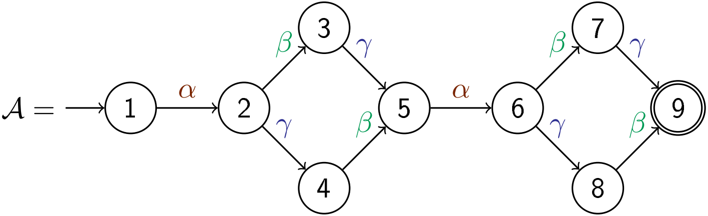
B : 1

C : 1

p_1 : 1

Actual state: 1

Distributed view



$\rho = \alpha$

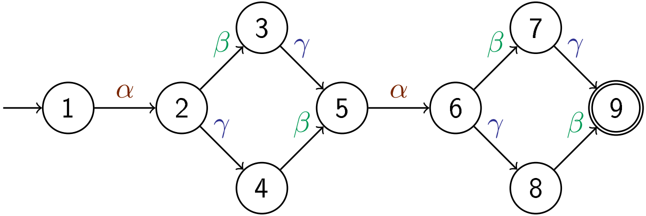
$$B : 1 \xrightarrow{\alpha} 2$$

$$C : 1 \xrightarrow{\alpha} 2$$

$$p_1 : 1$$

$$\text{Actual state: } 1 \xrightarrow{\alpha} 2$$

Distributed view

$\mathcal{A} =$ 

$\rho = \alpha \cdot \beta$

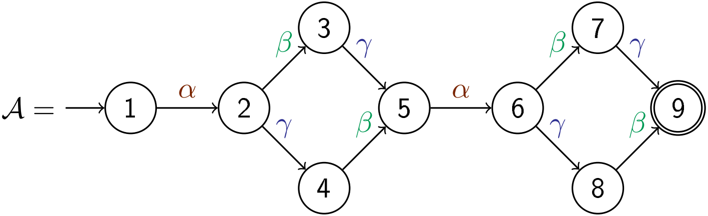
$$B : 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3$$

$$C : 1 \xrightarrow{\alpha} 2$$

$$p_1 : 1 \xrightarrow{\beta} 3$$

$$\text{Actual state: } 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3$$

Distributed view



$\rho = \alpha \cdot \beta \cdot \gamma$

$$B : 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3$$

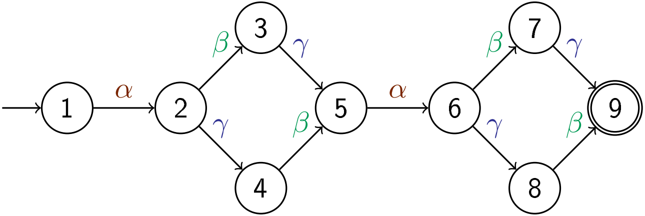
$$C : 1 \xrightarrow{\alpha} 2 \xrightarrow{\gamma} 4$$

$$p_1 : 1 \xrightarrow{\beta} 3$$

Actual state: $1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\gamma} 5$

Distributed view

$\mathcal{A} =$



$$\rho = \alpha \cdot \beta \cdot \gamma \cdot \alpha$$

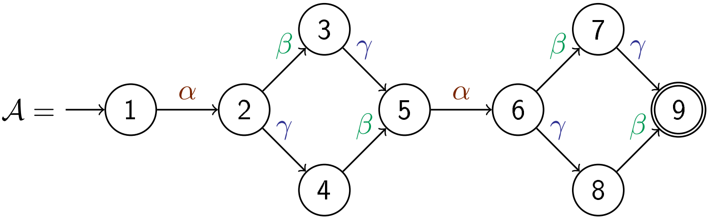
$$B : 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\alpha} 6$$

$$C : 1 \xrightarrow{\alpha} 2 \xrightarrow{\gamma} 4 \xrightarrow{\alpha} 6$$

$$p_1 : 1 \xrightarrow{\beta} 3$$

$$\text{Actual state: } 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\gamma} 5 \xrightarrow{\alpha} 6$$

Distributed view



$\rho = \alpha \cdot \beta \cdot \gamma \cdot \alpha \cdot \beta$

$$B : 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\alpha} 6 \xrightarrow{\beta} 7$$

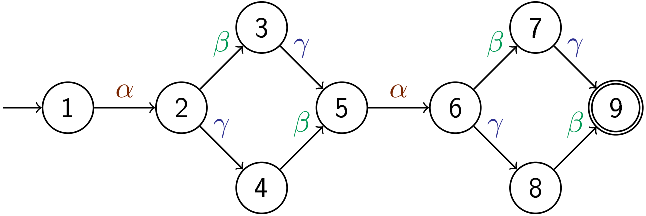
$$C : 1 \xrightarrow{\alpha} 2 \xrightarrow{\gamma} 4 \xrightarrow{\alpha} 6$$

$$p_1 : 1 \xrightarrow{\beta} 3 \xrightarrow{\beta} 7$$

$$\text{Actual state: } 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\gamma} 5 \xrightarrow{\alpha} 6 \xrightarrow{\beta} 7$$

Distributed view

$\mathcal{A} =$



$\rho = \alpha \cdot \beta \cdot \gamma \cdot \alpha \cdot \beta \cdot \gamma$

$$B : 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\alpha} 6 \xrightarrow{\beta} 7$$

$$C : 1 \xrightarrow{\alpha} 2 \xrightarrow{\gamma} 4 \xrightarrow{\alpha} 6 \xrightarrow{\gamma} 8$$

$$p_1 : 1 \xrightarrow{\beta} 3 \xrightarrow{\beta} 7$$

$$\text{Actual state: } 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\gamma} 5 \xrightarrow{\alpha} 6 \xrightarrow{\beta} 7 \xrightarrow{\gamma} 9$$

Zielonka's distributivity theorem

Theorem [Zielonka, 1987]

Every diamond-closed language can be recognized by an asynchronous automaton.

Zielonka's distributivity theorem

Theorem [Zielonka, 1987]

Every diamond-closed language can be recognized by an asynchronous automaton.

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

Zielonka's distributivity theorem

Theorem [Zielonka, 1987]

Every diamond-closed language can be recognized by an asynchronous automaton.

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

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)

- Adapt Zielonka's result to this setting:
  - Input language contains instructions for reconfiguration
  - Output automaton should implement them only with local information

Our result

Theorem

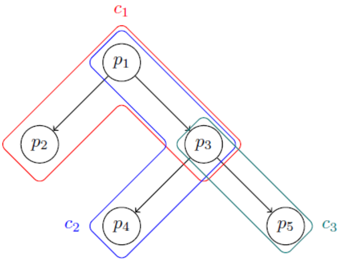
Every diamond-closed language with reconfiguration operations[2] and over a tree-like communication architecture[1] can be recognized by a reconfigurable[3] asynchronous automaton.

Our result

Theorem

Every diamond-closed language with reconfiguration operations[2] and over a tree-like communication architecture[1] can be recognized by a reconfigurable[3] asynchronous automaton.

1 :

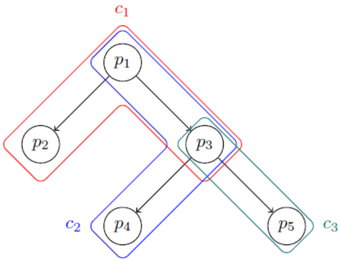


Our result

Theorem

Every diamond-closed language with reconfiguration operations[2] and over a tree-like communication architecture[1] can be recognized by a reconfigurable[3] asynchronous automaton.

1 :



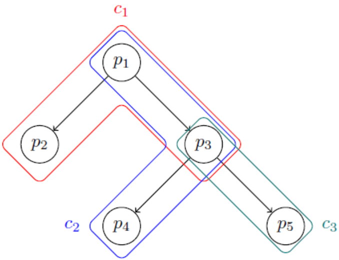
2 : connecting, disconnecting, local change of position in the tree

Our result

Theorem

Every diamond-closed language with reconfiguration operations[2] and over a tree-like communication architecture[1] can be recognized by a reconfigurable[3] asynchronous automaton.

1 :



2 : connecting, disconnecting, local change of position in the tree

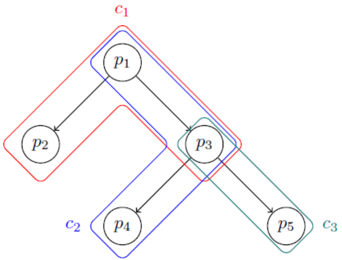
3 : set of channels listened to depends on state of process

Our result

Theorem

Every diamond-closed language with reconfiguration operations[2] and over a tree-like communication architecture[1] can be recognized by a reconfigurable[3] asynchronous automaton.

1 :



2 : connecting, disconnecting, local change of position in the tree

3 : set of channels listened to depends on state of process

- Complexity: $O(|\mathcal{A}|^2 \cdot 2^{|\mathrm{CH}|})$

Conclusion

Summary

Adapted tree construction for Zielonka's distributivity theorem to the reconfigurable setting.

Conclusion

Summary

Adapted tree construction for Zielonka's distributivity theorem to the reconfigurable setting.

Future works

Get rid of the tree restriction! (not easy...)

Conclusion

Summary

Adapted tree construction for Zielonka's distributivity theorem to the reconfigurable setting.

Future works

Get rid of the tree restriction! (not easy...)

Thanks, questions?