Adding Reconfiguration to Zielonka's Asynchronous Automata

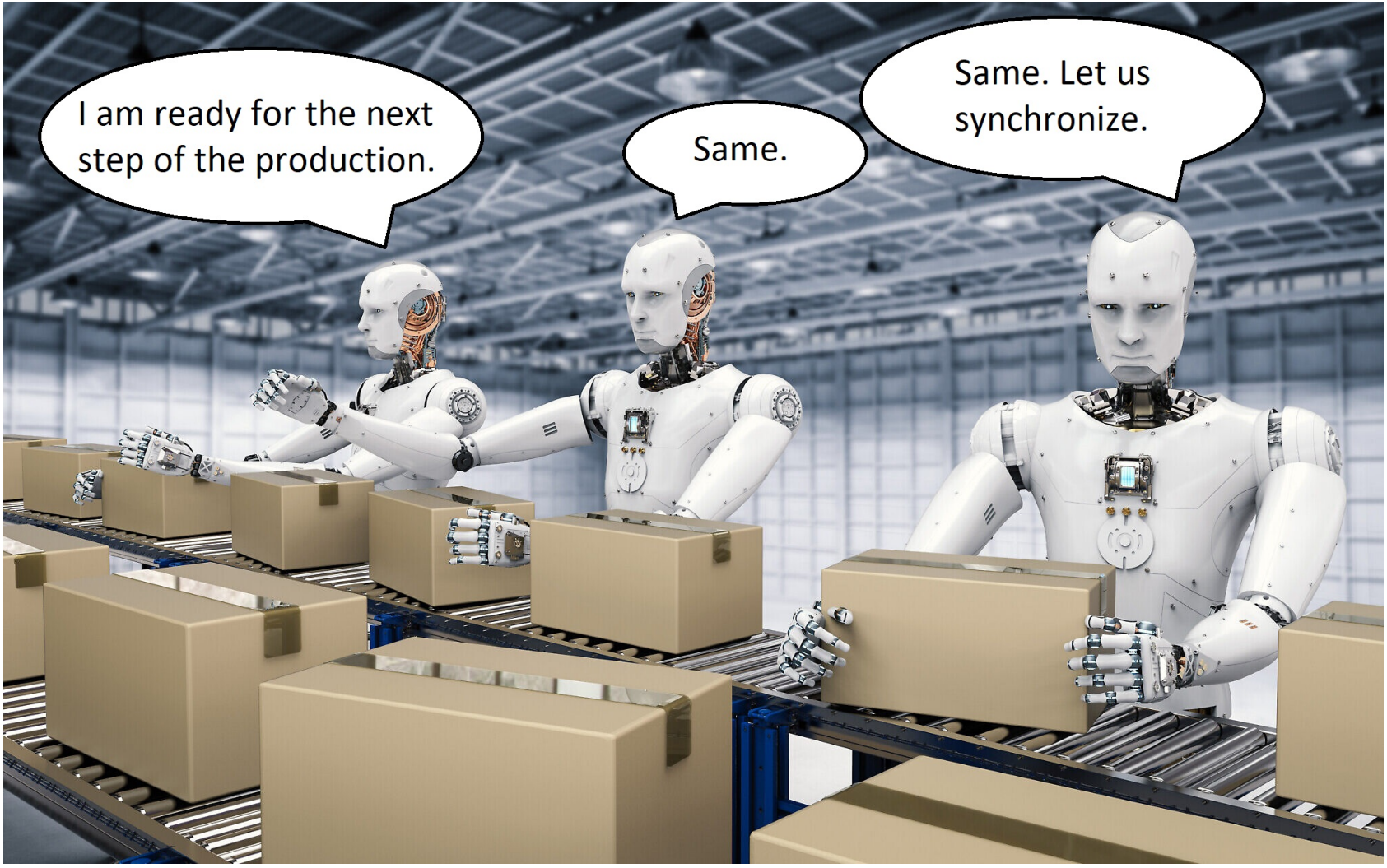
Mathieu Lehaut, Nir Piterman

University of Gothenburg, Sweden

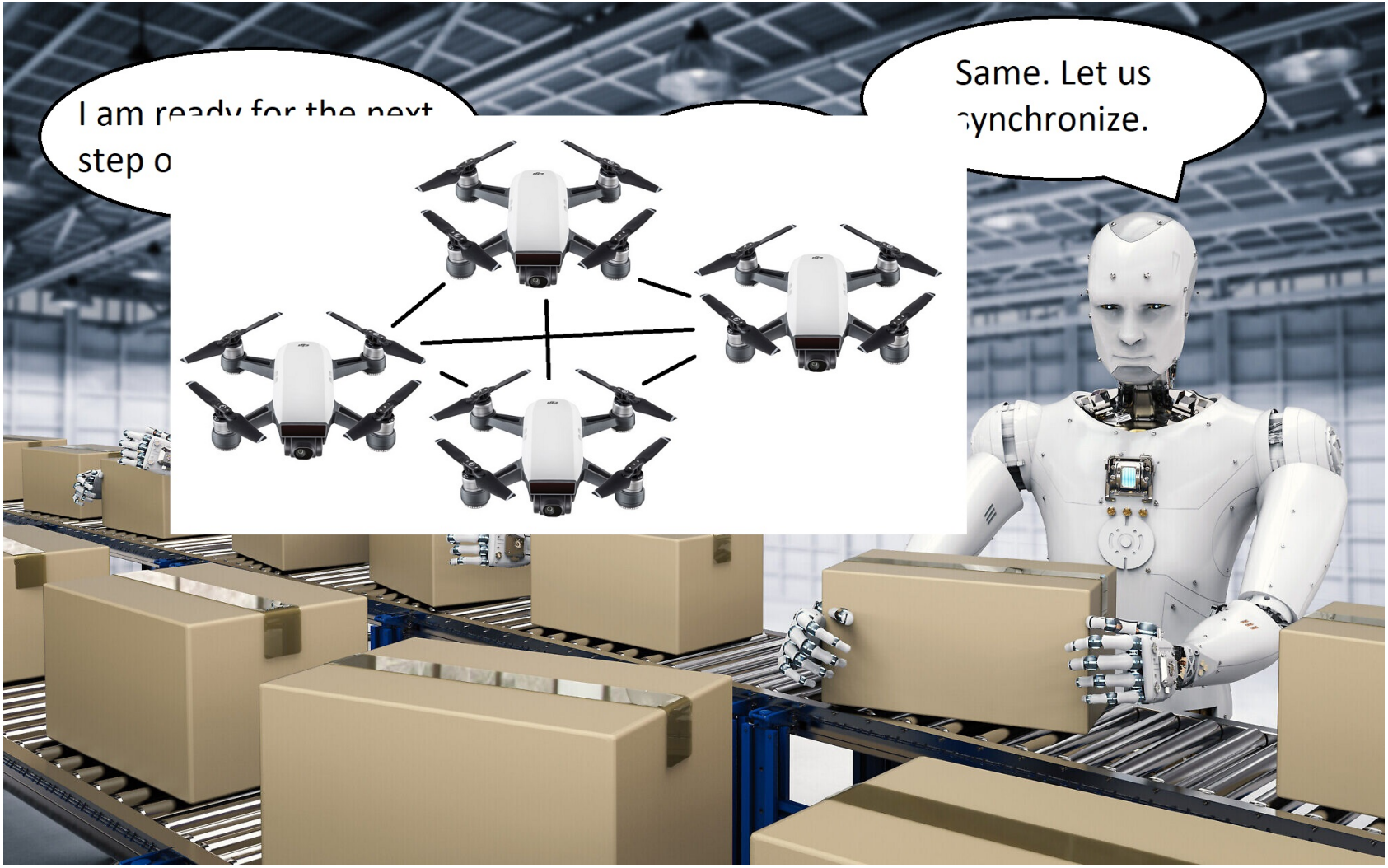
GandALF 2024

Motivation

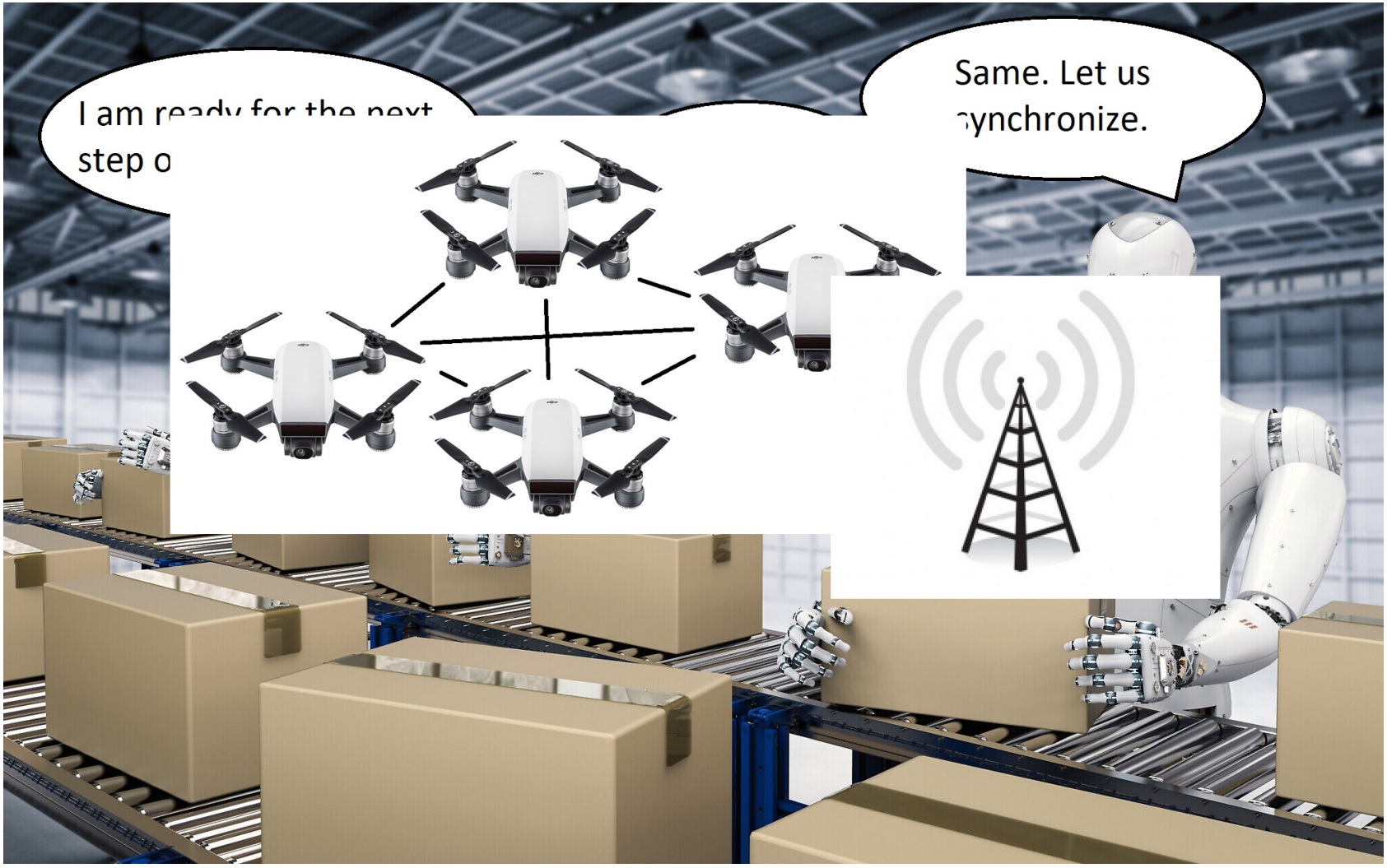
Motivation


I am ready for the next step of the production.
Same.
Same. Let us synchronize.

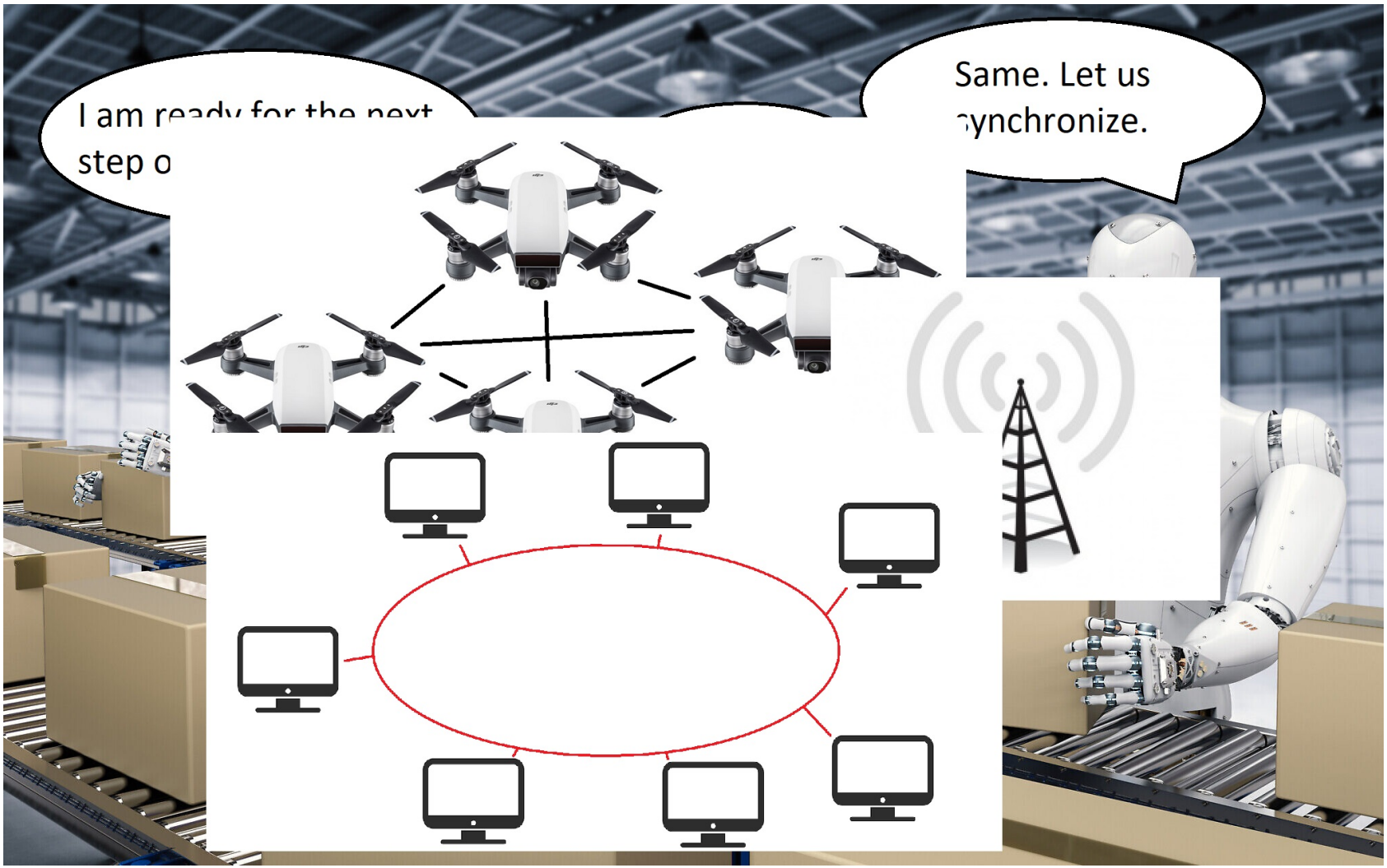
Motivation



Motivation

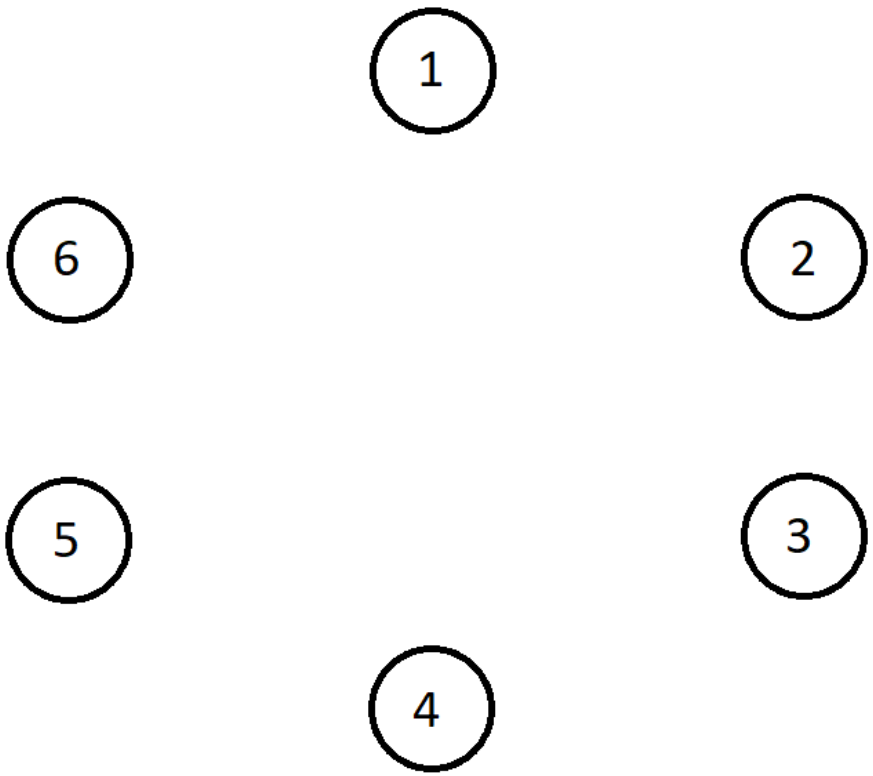


Motivation



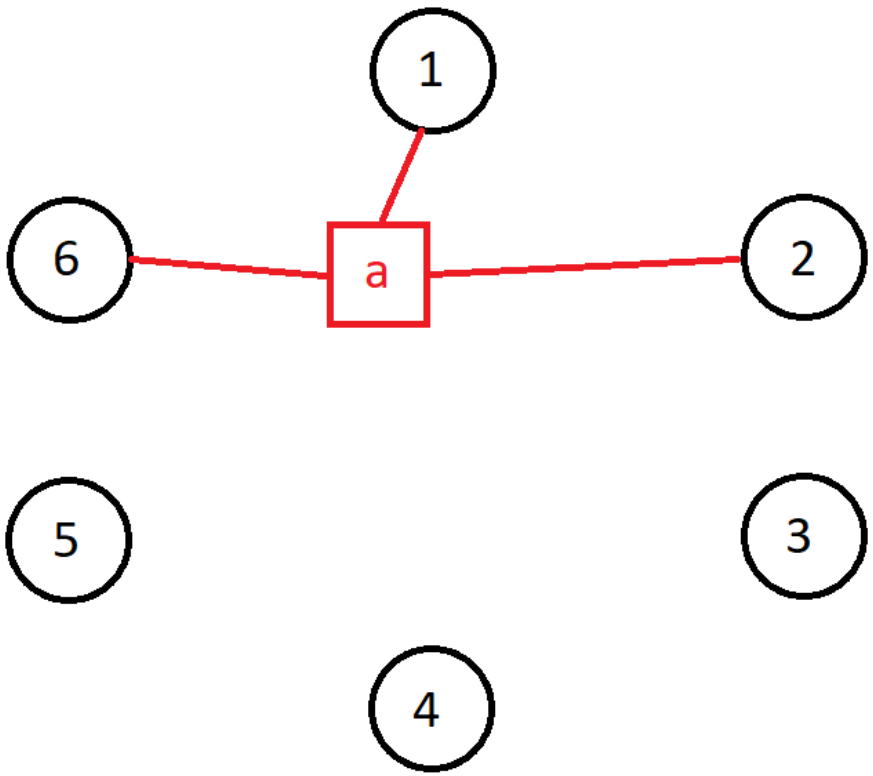
Distributed setting

Independent processes



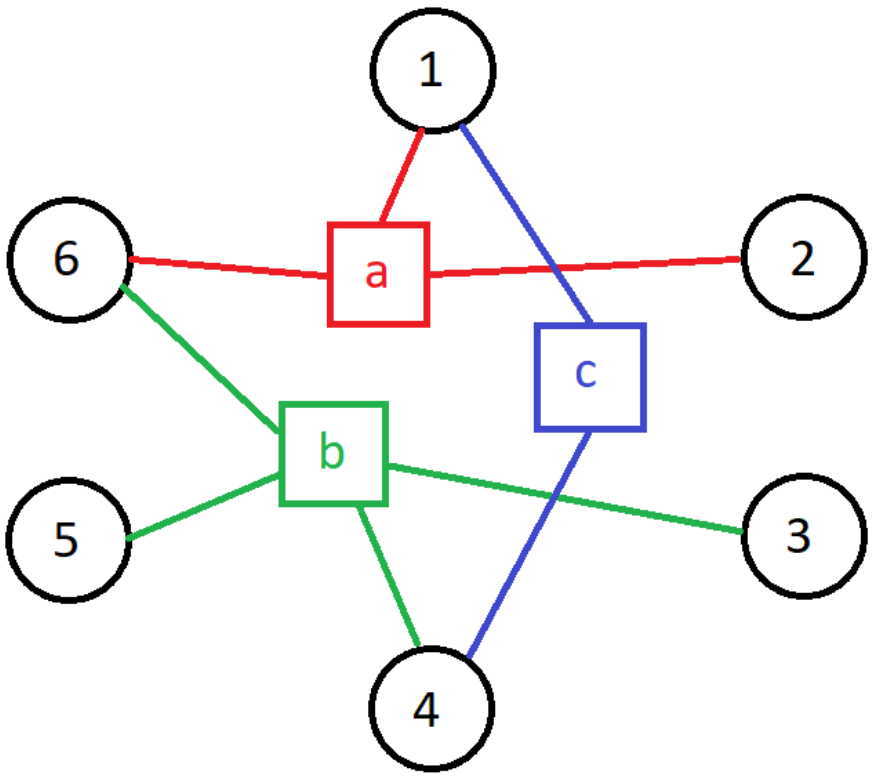
Distributed setting

Independent processes communicating over channels



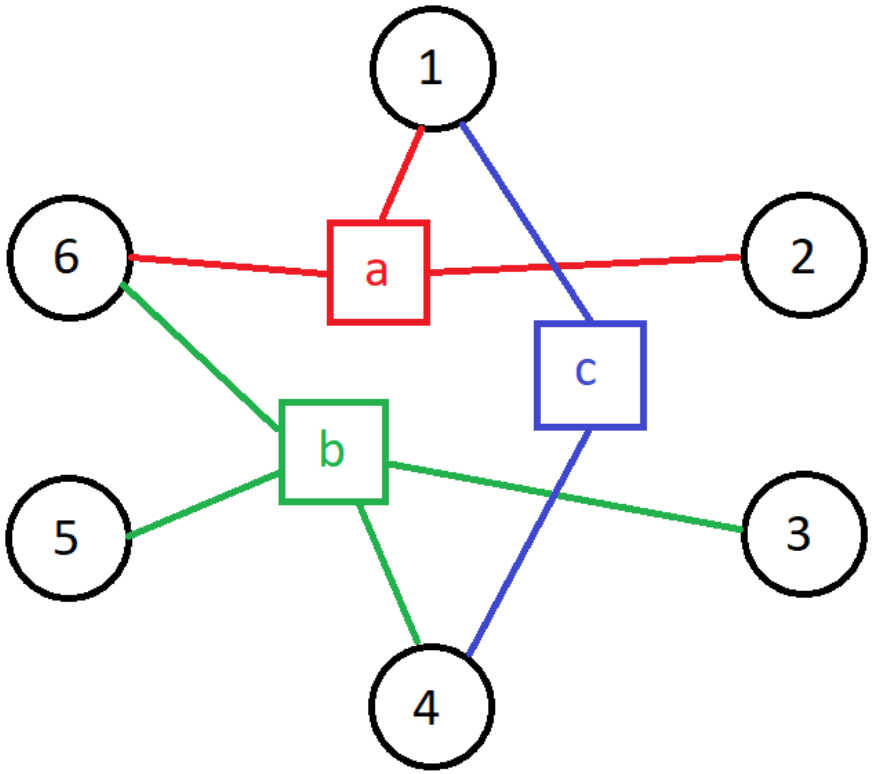
Distributed setting

Independent processes communicating over channels



Distributed setting

Independent processes communicating over channels

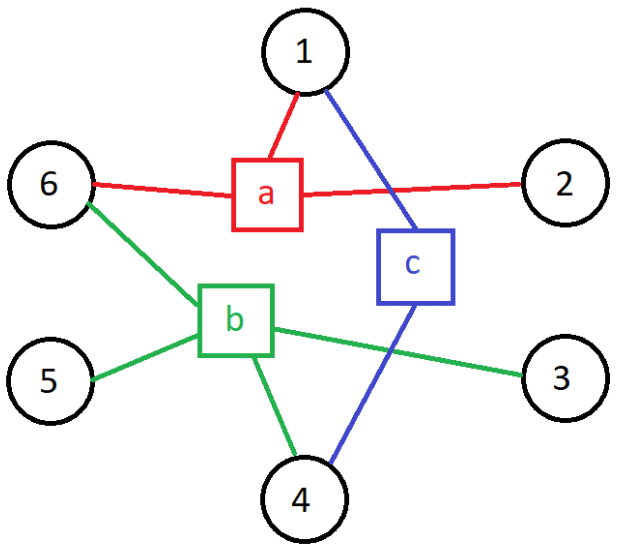


Some assumptions

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

Fixed or reconfigurable communications

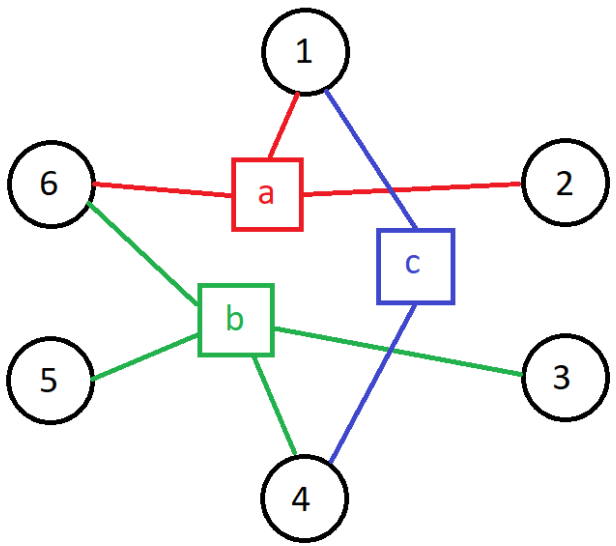
Fixed topology:



$1 : a, c \quad 2 : a \quad 3 : b \quad 4 : b, c \quad 5 : b \quad 6 : a, b$

Fixed or reconfigurable communications

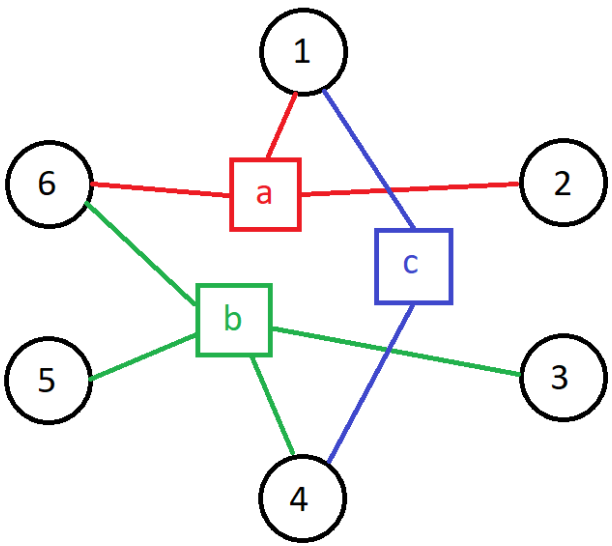
With reconfiguration: before...

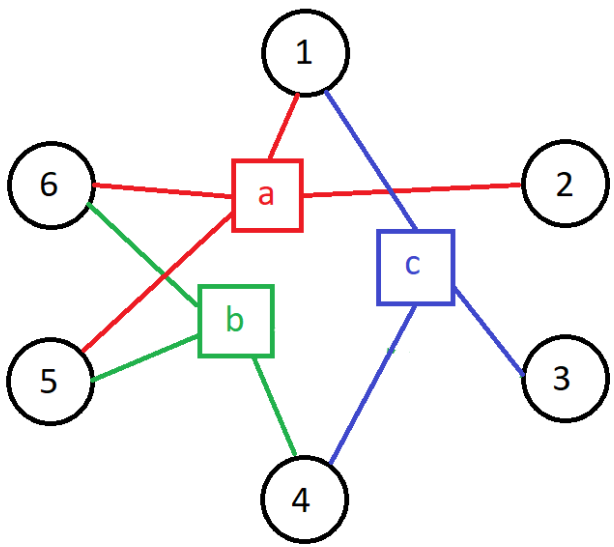


$1 : a, c \quad 2 : a \quad 3 : b$
$4 : b, c \quad 5 : b \quad 6 : a, b$

Fixed or reconfigurable communications

With reconfiguration: before... ... and after





$1 : a, c \quad 2 : a \quad 3 : b$
$4 : b, c \quad 5 : b \quad 6 : a, b$

$1 : a, c \quad 2 : a \quad 3 : c$
$4 : b, c \quad 5 : a, b \quad 6 : a, b$

Fixed communication model

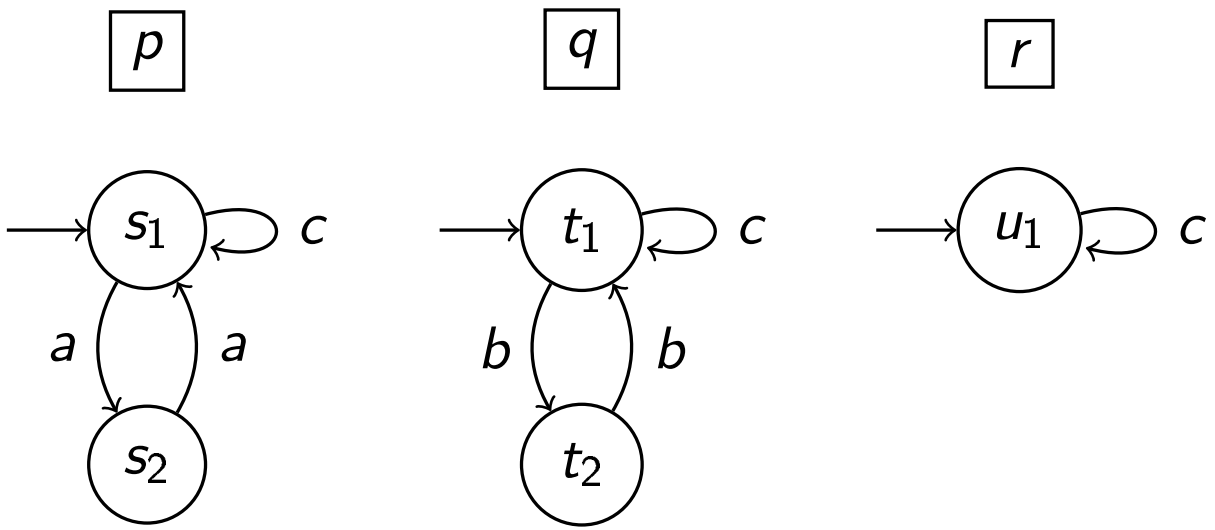
Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

Fixed communication model

Zielonka's Asynchronous Automata (AA)

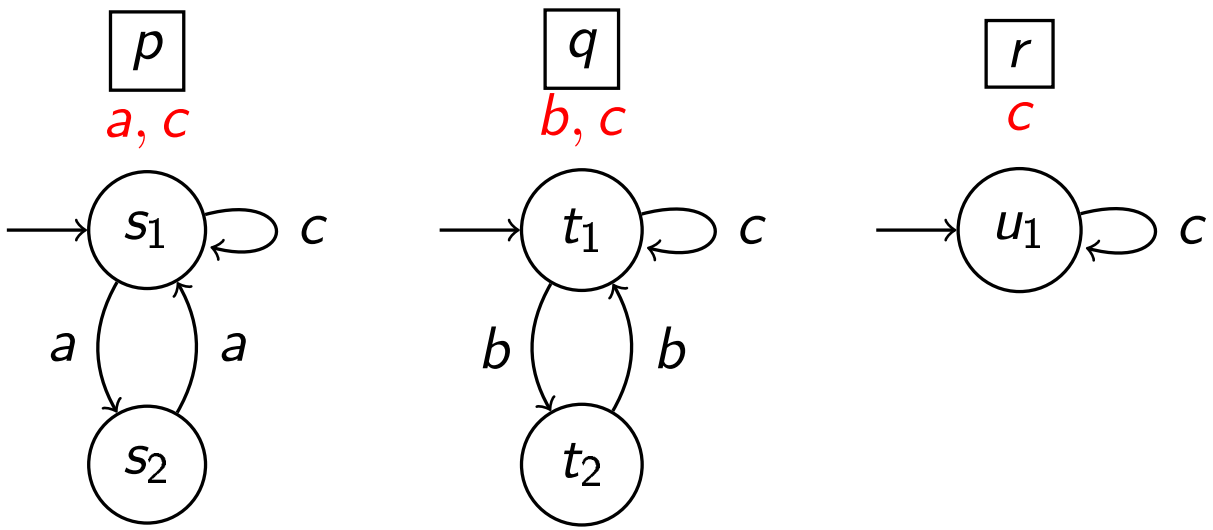
Fix set of processes, channels, and communication structure



Fixed communication model

Zielonka's Asynchronous Automata (AA)

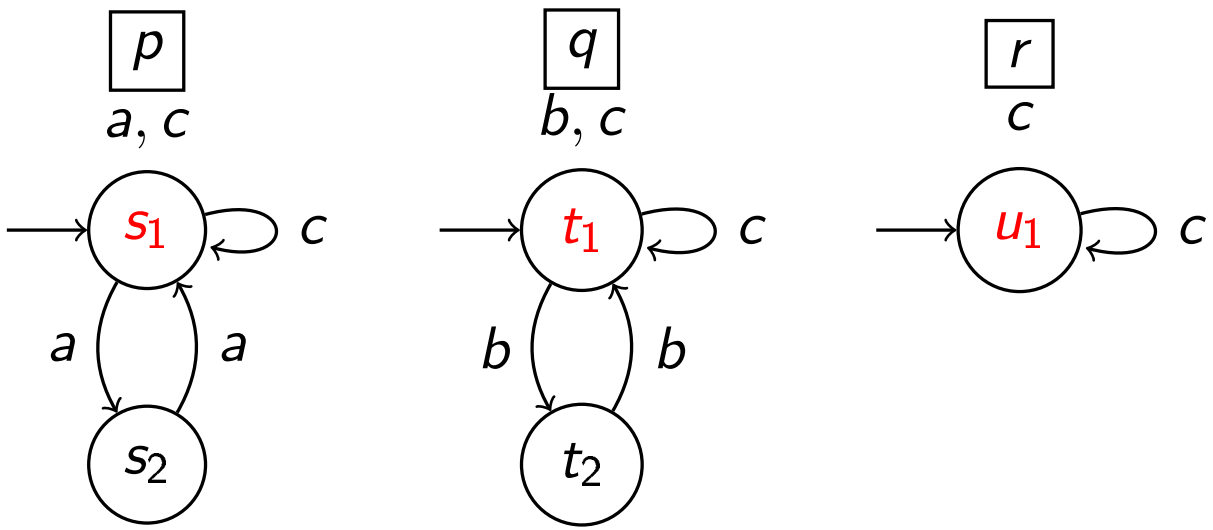
Fix set of processes, channels, and communication structure



Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

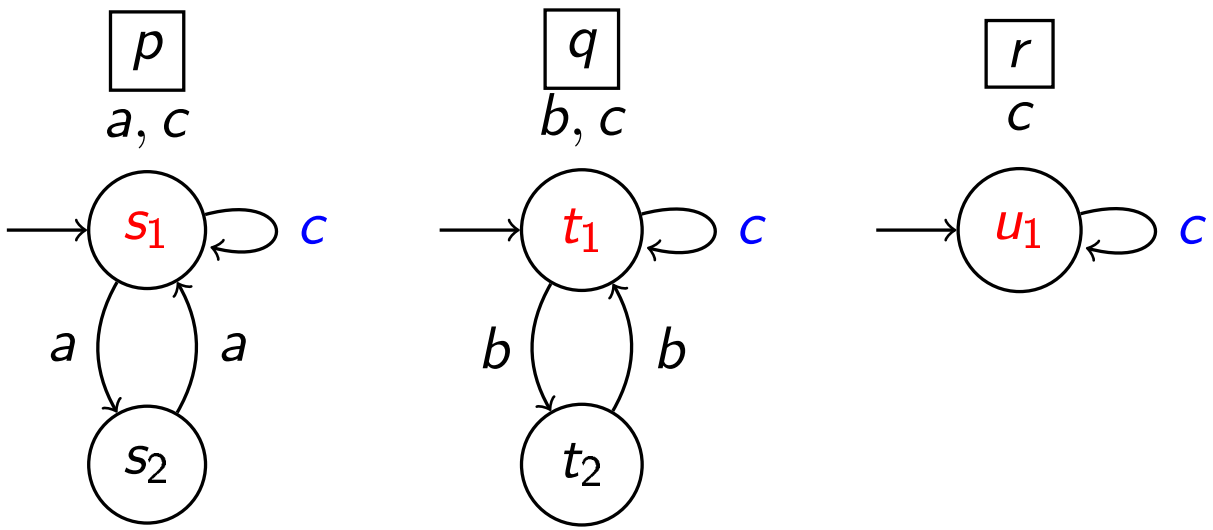


$\rho = (s_1, t_1, u_1)$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

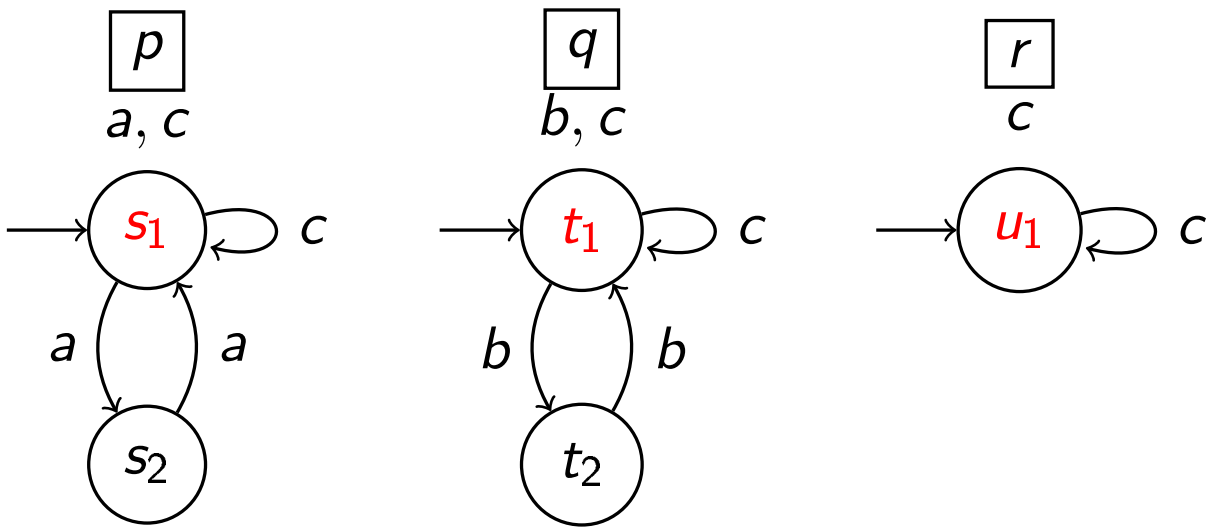


$\rho = (s_1, t_1, u_1) \rightarrow^c$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

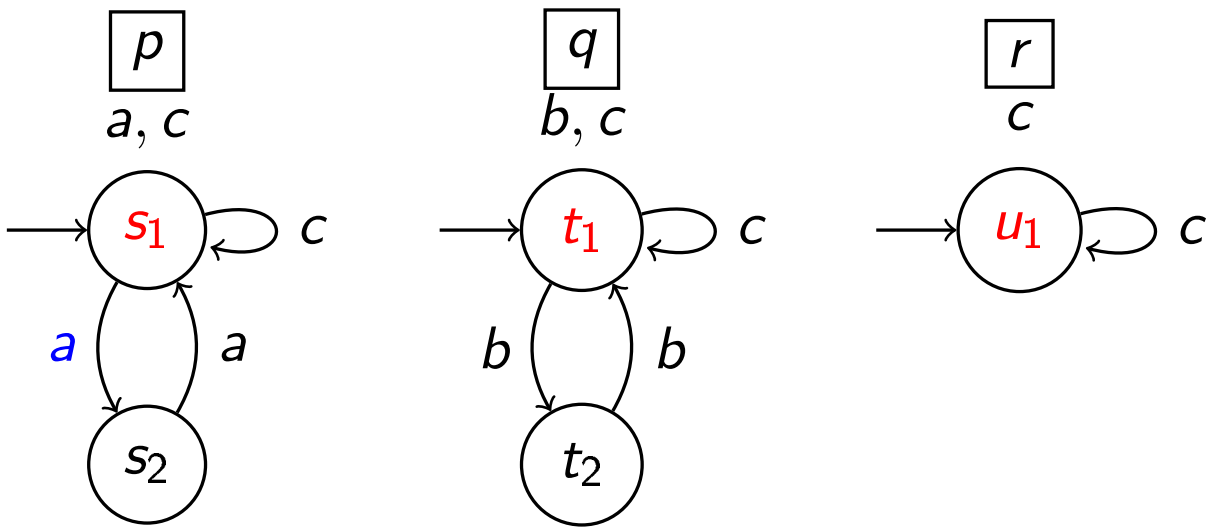


$\rho = (s_1, t_1, u_1) \rightarrow^c (s_1, t_1, u_1)$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

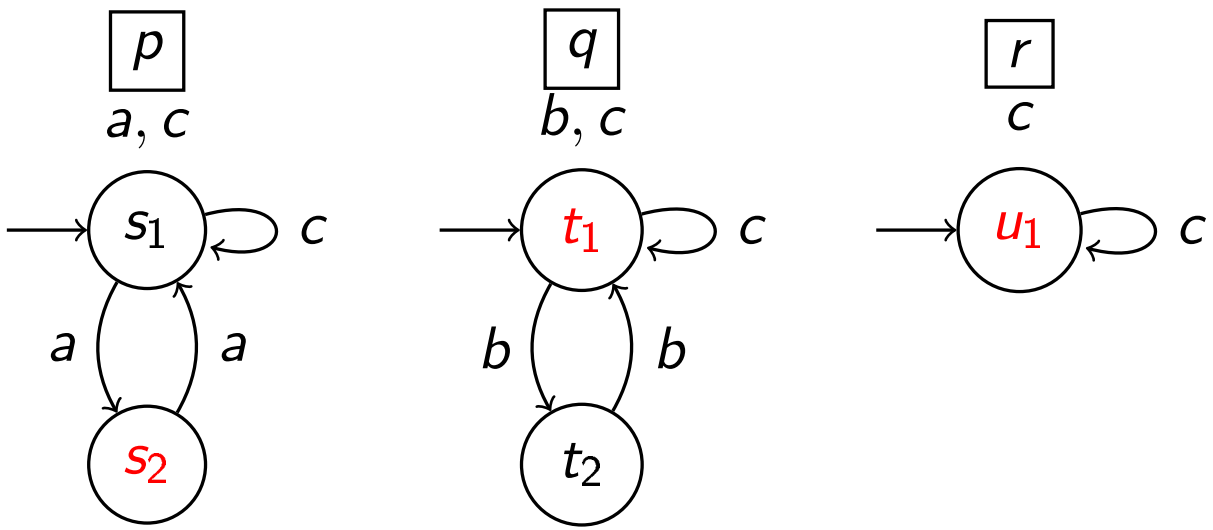


$$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1) \to^{a}$$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

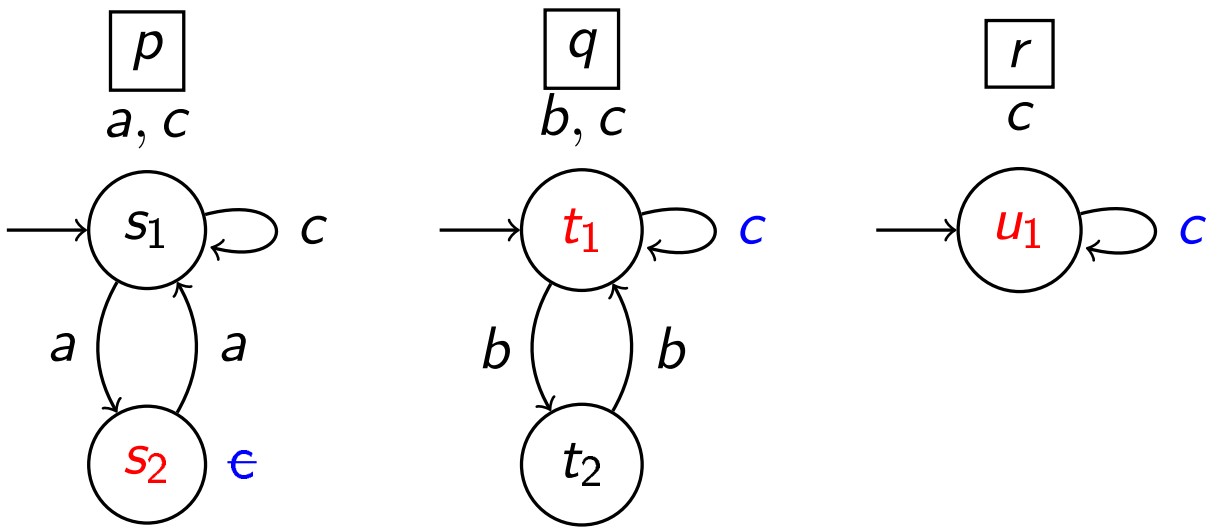


$$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1) \to^{a} (s_2, t_1, u_1)$$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

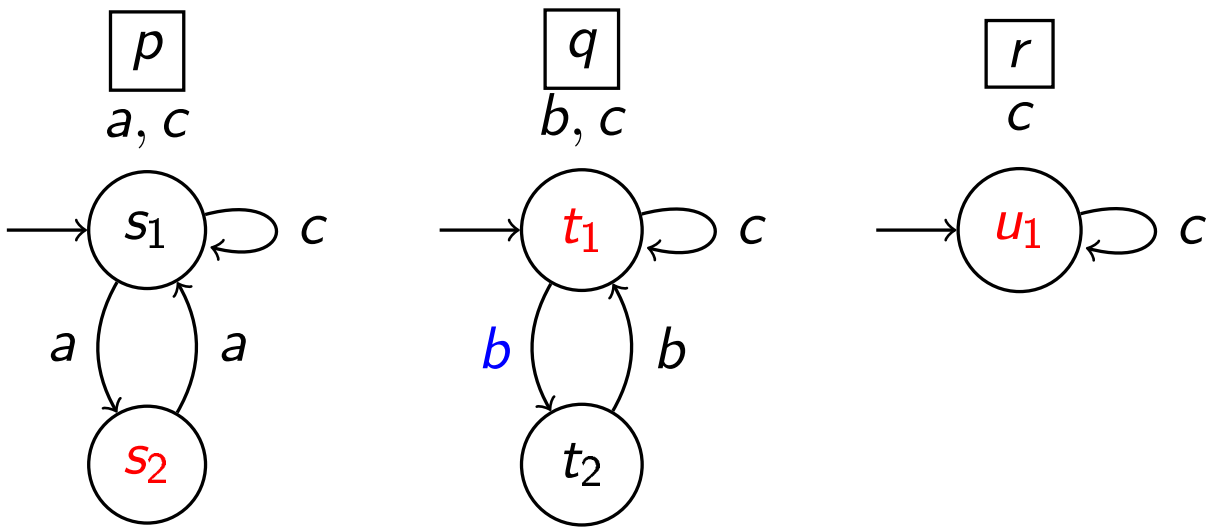


$$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1) \to^{a} (s_2, t_1, u_1) \not\to^{c}$$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure



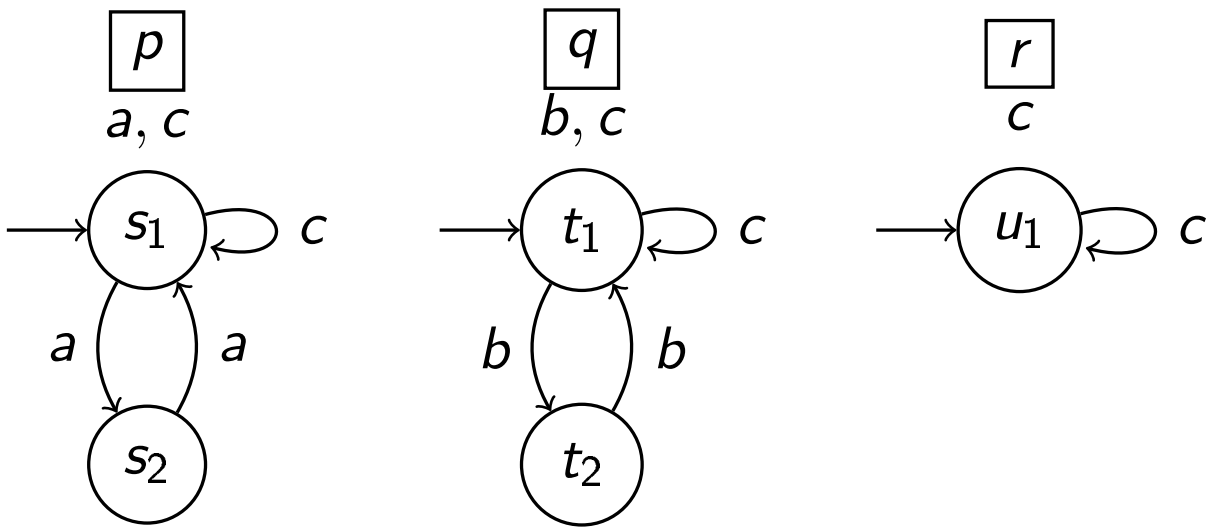
$\rho = (s_1, t_1, u_1) \to^c (s_1, t_1, u_1) \to^a (s_2, t_1, u_1) \to^b \ldots$

$w = cab\ldots$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure



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

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

Reconfigurable model

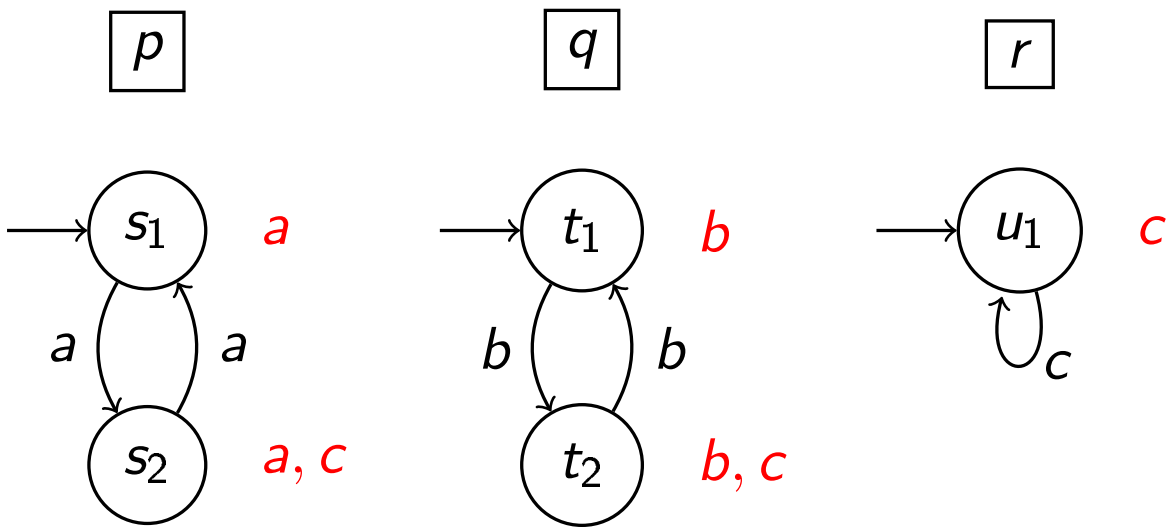
Reconfigurable Asynchronous Automata (RAA)

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

Reconfigurable model

Reconfigurable Asynchronous Automata (RAA)

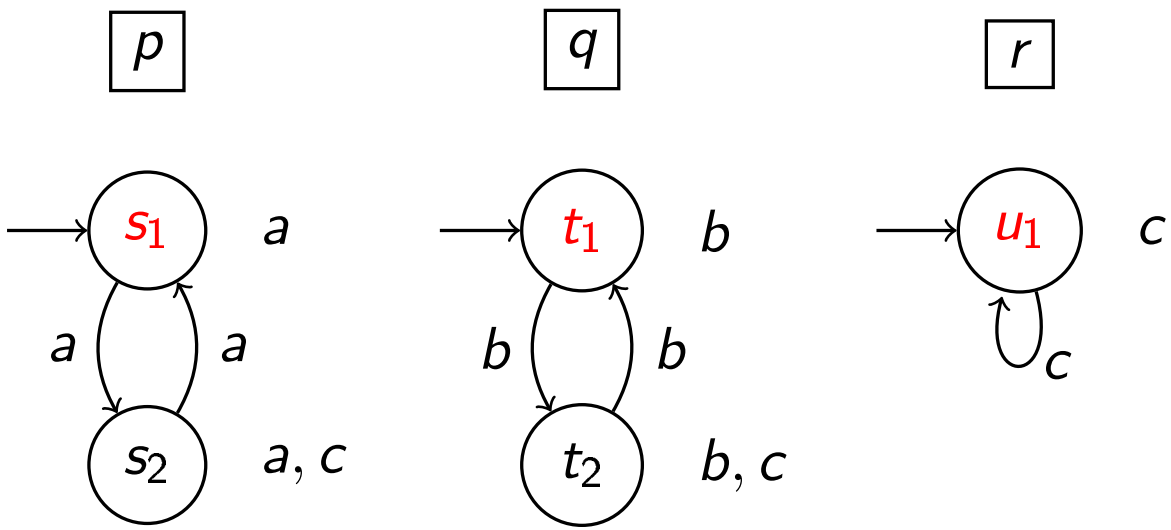
Fix set of processes, channels, and ~~communication structure~~
Communication structure is state-dependent



Reconfigurable model

Reconfigurable Asynchronous Automata (RAA)

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

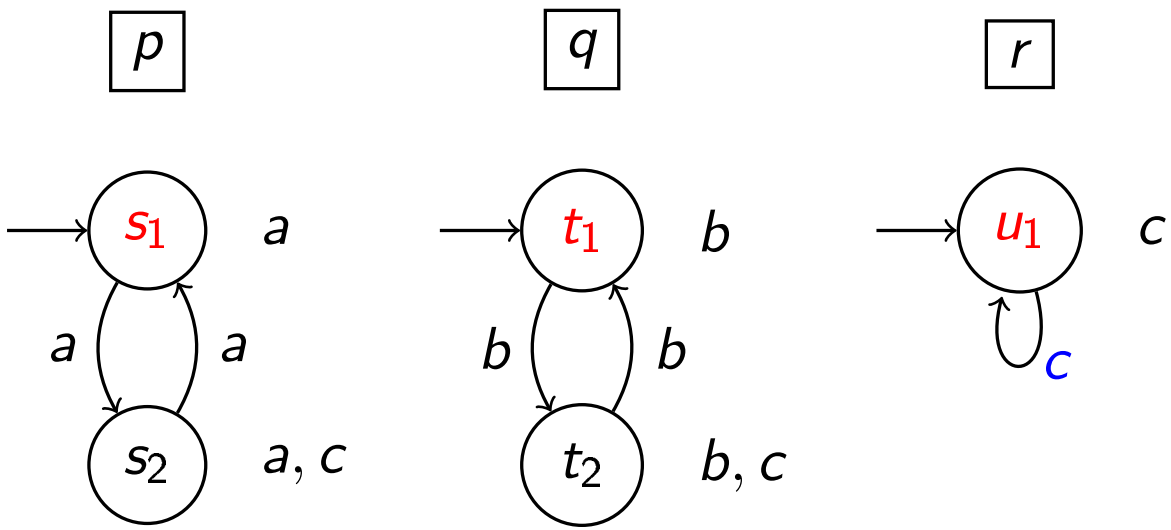


$\rho = (s_1, t_1, u_1)$

Reconfigurable model

Reconfigurable Asynchronous Automata (RAA)

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

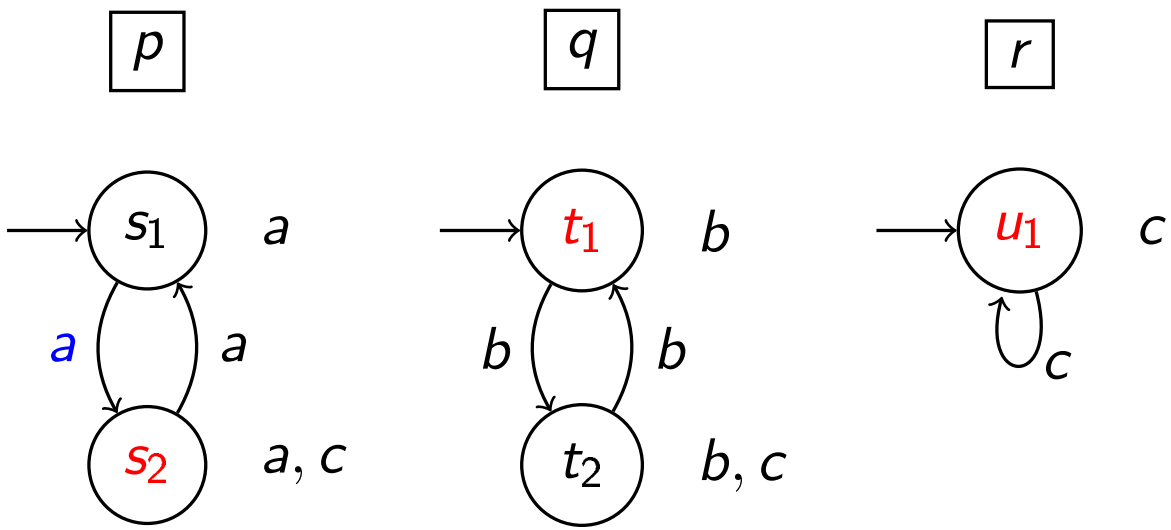


$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1)$

Reconfigurable model

Reconfigurable Asynchronous Automata (RAA)

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

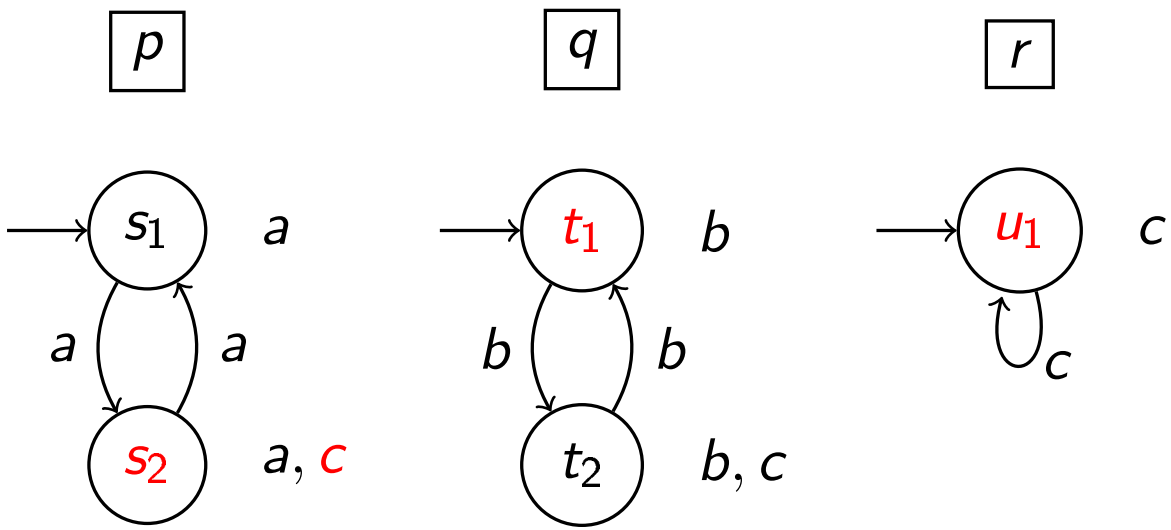


$$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1) \to^{a} (s_2, t_1, u_1)$$

Reconfigurable model

Reconfigurable Asynchronous Automata (RAA)

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



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

Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

Any AA can be transformed into a RAA with the same language, same channels, and same processes

Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

Any AA can be transformed into a RAA with the same language, same channels, and same processes

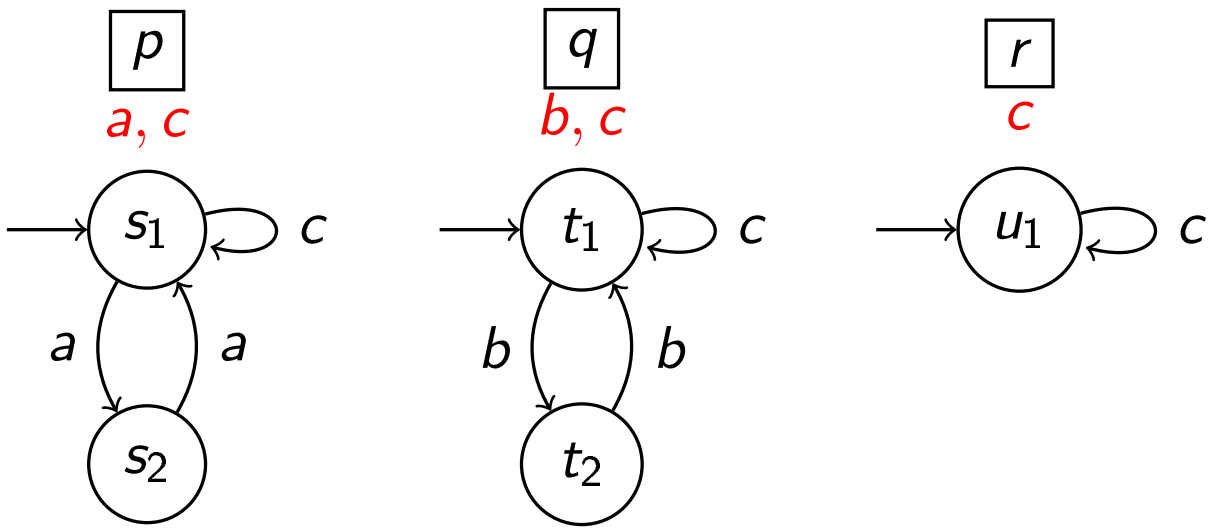
Proof: Always listen to the same (fixed) set of channels in all states

Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

Any AA can be transformed into a RAA with the same language, same channels, and same processes

Proof: Always listen to the same (fixed) set of channels in all states

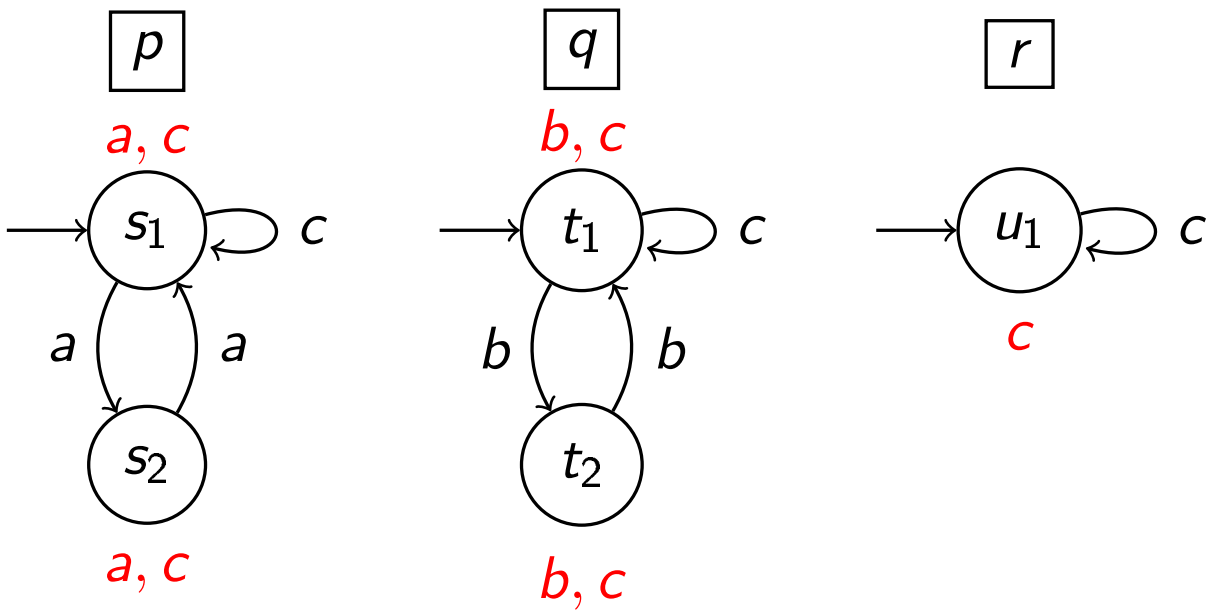


Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

Any AA can be transformed into a RAA with the same language, same channels, and same processes

Proof: Always listen to the same (fixed) set of channels in all states



Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

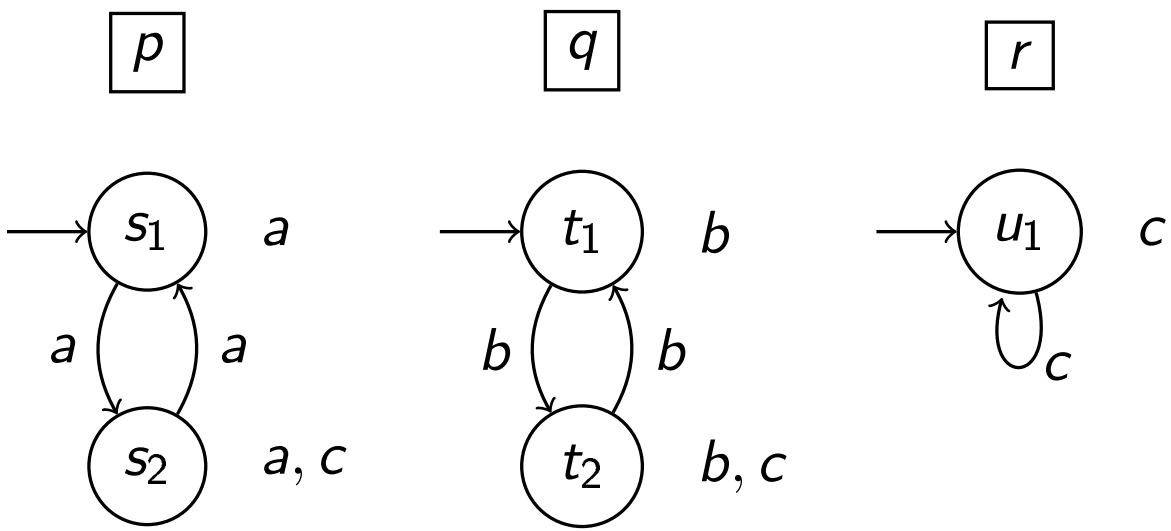
Proof idea: Listen to every channel, ignore (but don't block!) communications you were not supposed to listen to originally

Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

Proof idea: Listen to every channel, ignore (but don't block!) communications you were not supposed to listen to originally

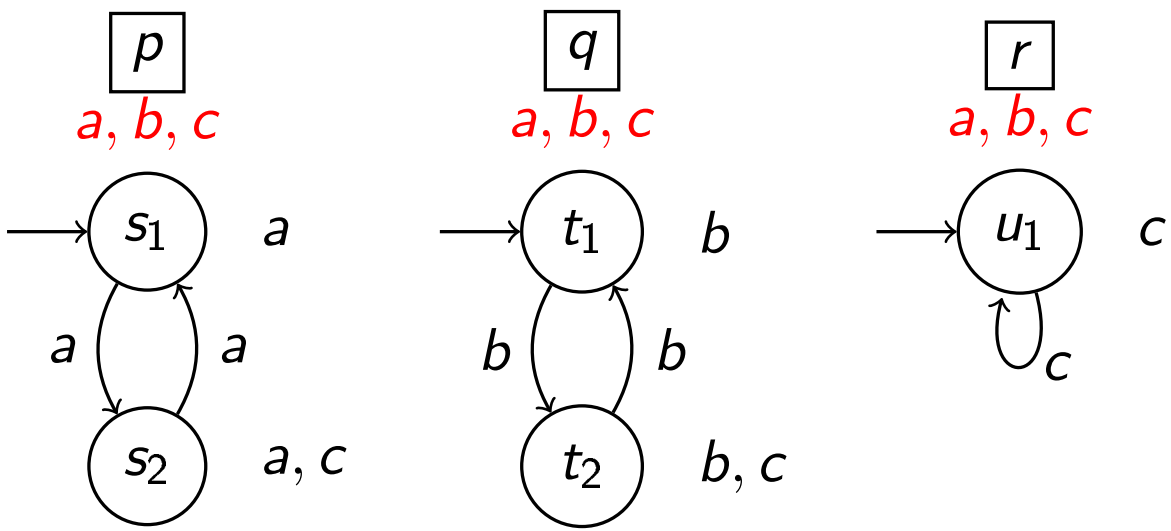


Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

Proof idea: Listen to every channel, ignore (but don't block!) communications you were not supposed to listen to originally

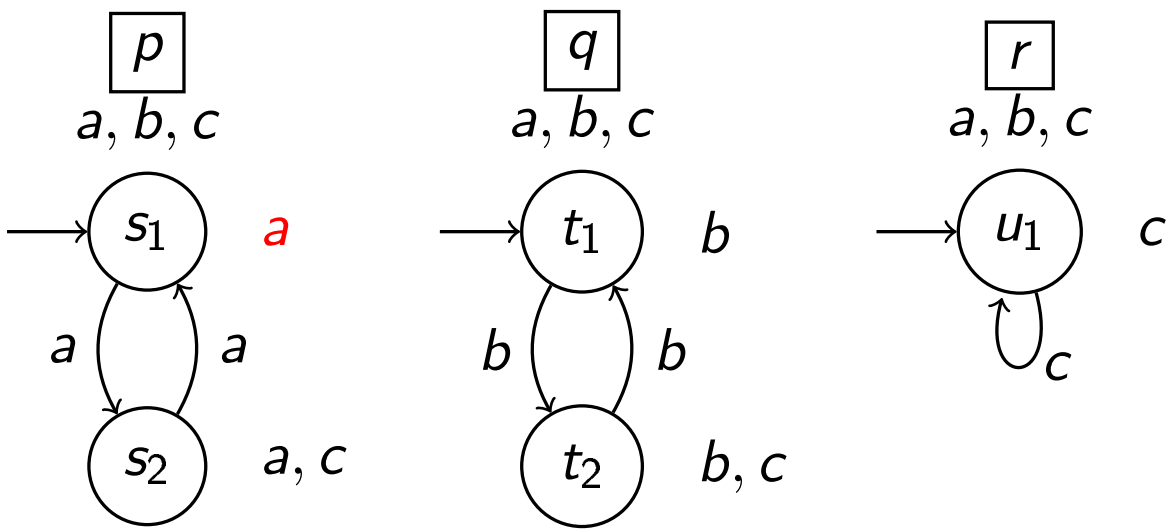


Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

Proof idea: Listen to every channel, ignore (but don't block!) communications you were not supposed to listen to originally

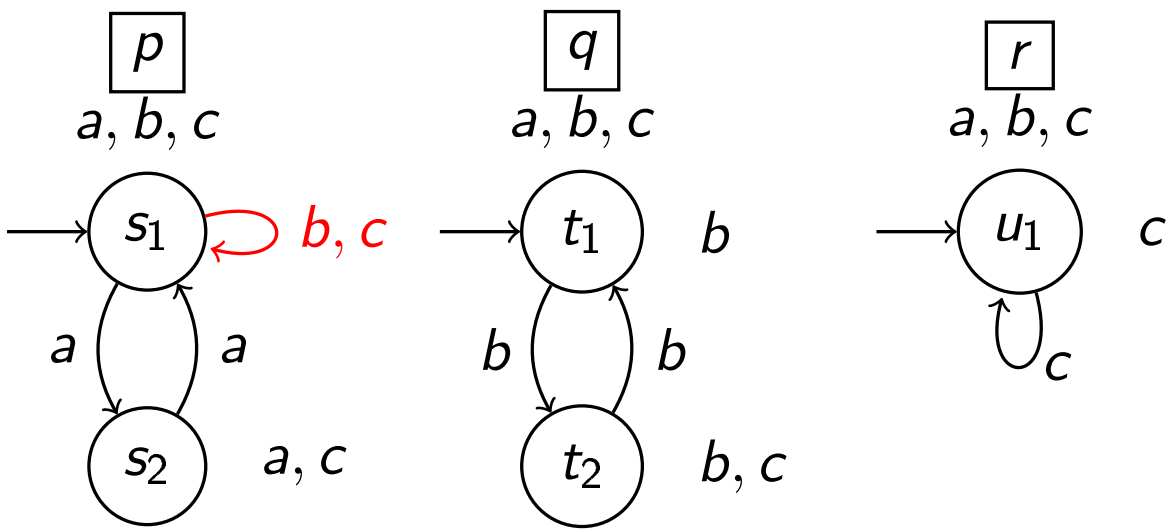


Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

Proof idea: Listen to every channel, ignore (but don't block!) communications you were not supposed to listen to originally

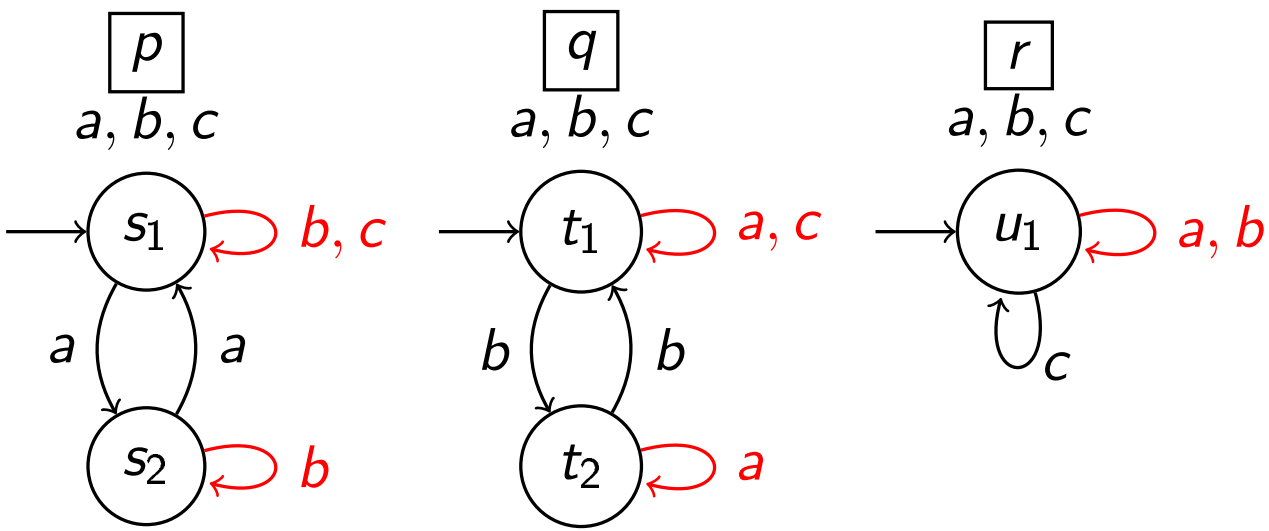


Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

Proof idea: Listen to every channel, ignore (but don't block!) communications you were not supposed to listen to originally



Now what?

Adding reconfigurability does not increase expressiveness.

Now what?

Adding reconfigurability does not increase expressiveness.

But...

Now what?

Adding reconfigurability does not increase expressiveness.

But...

Previous construction is unsatisfactory:
- All processes listen to all channels

Now what?

Adding reconfigurability does not increase expressiveness.

But...

Previous construction is unsatisfactory:
- All processes listen to all channels

Can we do better?

Now what?

Adding reconfigurability does not increase expressiveness.

But...

Previous construction is unsatisfactory:
All processes listen to all channels

Can we do better? Not really...

Result 3: No better general construction

There is a RAA such that in any equivalent AA, every process:

- either listens to every channel, or
- can be lead to a passive all-accepting state.

Proof idea

Main ideas

- We build an RAA with n processes and $n+1$ channels.
- The behavior (blocking/allowing communications) of a process changes when receiving instructions from a special *switching channel*.

Proof idea

Main ideas

- We build an RAA with n processes and $n+1$ channels.
- The behavior (blocking/allowing communications) of a process changes when receiving instructions from a special *switching channel*.
- After enough communications, a different channel assumes the role of the switching channel, and so on.

Proof idea

Main ideas

- We build an RAA with n processes and $n+1$ channels.
- The behavior (blocking/allowing communications) of a process changes when receiving instructions from a special *switching channel*.
- After enough communications, a different channel assumes the role of the switching channel, and so on.
- With a fixed topology, not listening to the switching channel means accepting everything.

More results about this construction

Previous statement

There is a RAA such that in any equivalent AA, every process:

- either listens to every channel, or
- can be lead to a passive all-accepting state.

More results about this construction

Strenghtened statement

There is a RAA such that in any equivalent AA there is an alternative initial configuration from which the same language is accepted, and every process:

- either listens to every channel, or
- starts in a passive all-accepting state.

More results about this construction

Strenghtened statement

There is a RAA such that in any equivalent AA there is an alternative initial configuration from which the same language is accepted, and every process:

- either listens to every channel, or
- starts in a passive all-accepting state.

"Complexity"

Every process in the RAA listens to at most 3 channels at a time ($O(1)$ connections).
Every non-trivial process in the AA listens to all channels ($O(n)$ connections).

Summary and future works

Takeaway message

- Adding reconfigurability does not increase expressiveness, but...

Summary and future works

Takeaway message

- Adding reconfigurability does not increase expressiveness, but...
- It may significantly reduce the number of connections needed.

Summary and future works

Takeaway message

- Adding reconfigurability does not increase expressiveness, but...
- It may significantly reduce the number of connections needed.

What next?

- Extend Zielonka's distributability theorem to reconfigurable model (work on tree topologies)

Summary and future works

Takeaway message

- Adding reconfigurability does not increase expressiveness, but...
- It may significantly reduce the number of connections needed.

What next?

- Extend Zielonka's distributability theorem to reconfigurable model (work on tree topologies)
- How to measure (then minimize) the amount of communications in a system while keeping the same behaviors?

Summary and future works

Takeaway message

- Adding reconfigurability does not increase expressiveness, but...
- It may significantly reduce the number of connections needed.

What next?

- Extend Zielonka's distributability theorem to reconfigurable model (work on tree topologies)
- How to measure (then minimize) the amount of communications in a system while keeping the same behaviors?

Thank you for your attention!

The switching RAA for $n = 2$