Understanding the Gain of Reconfigurable Communication

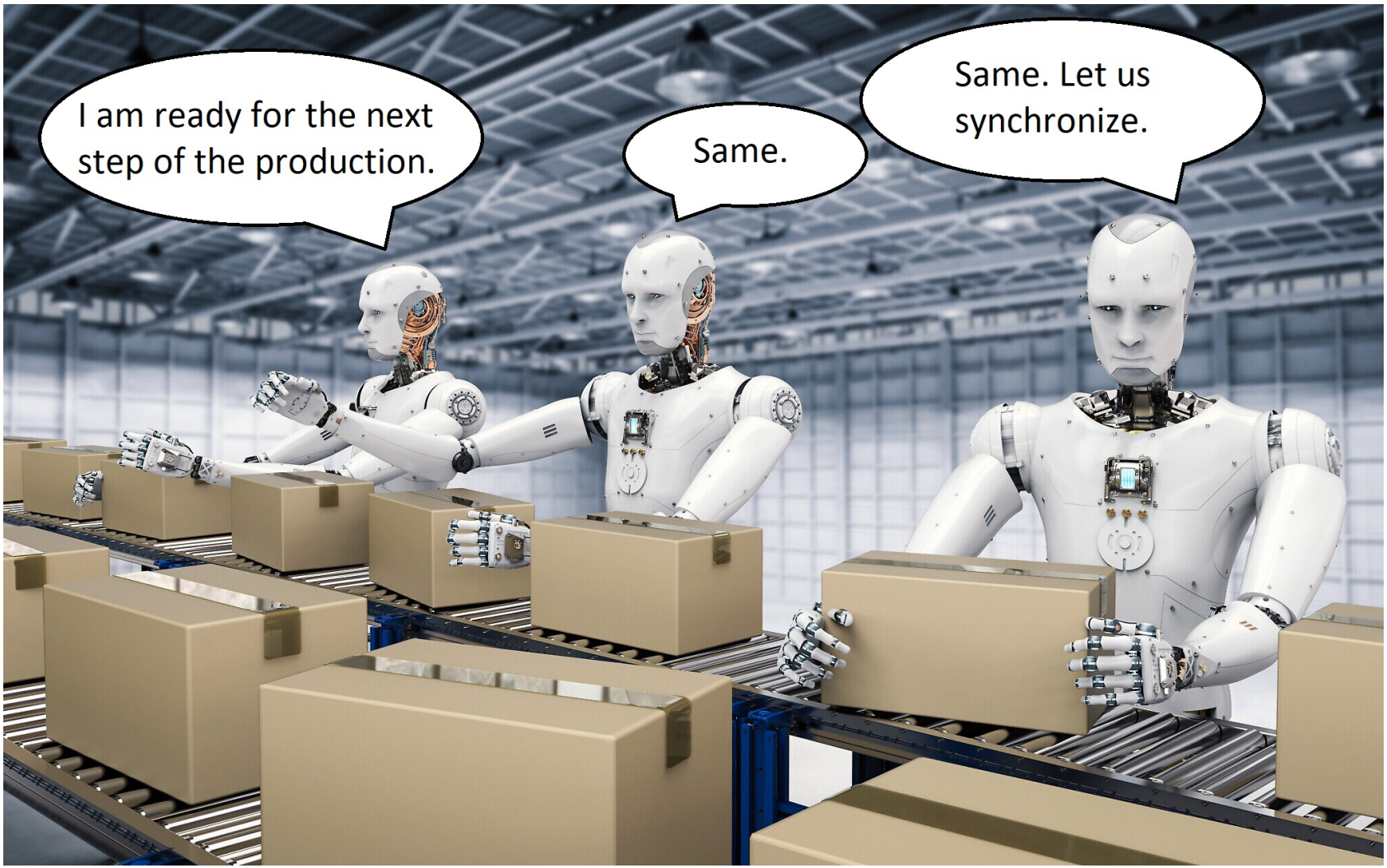
Mathieu Lehaut, Nir Piterman

University of Gothenburg, Sweden

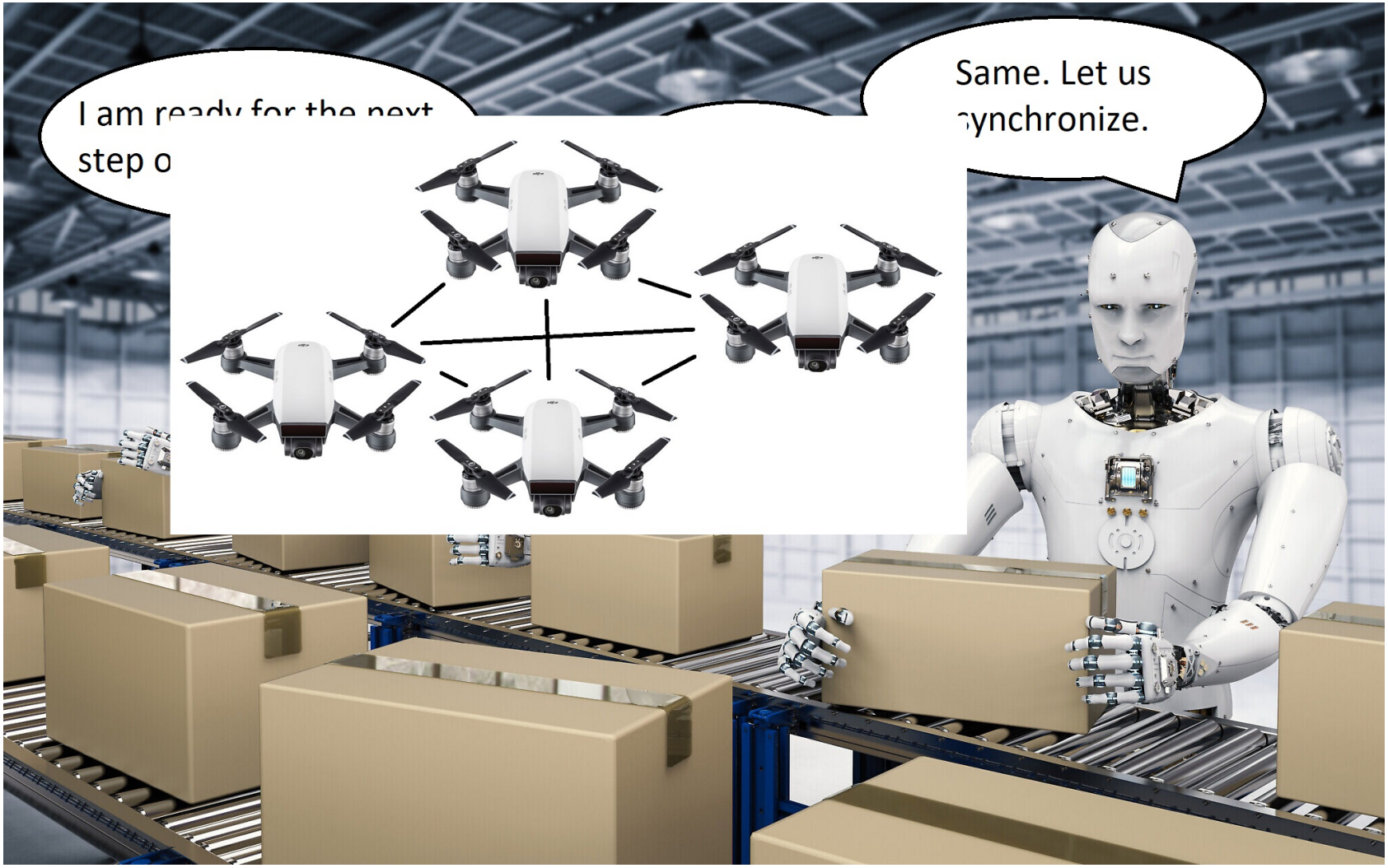
YR-CONCUR Workshop 2023

Motivation

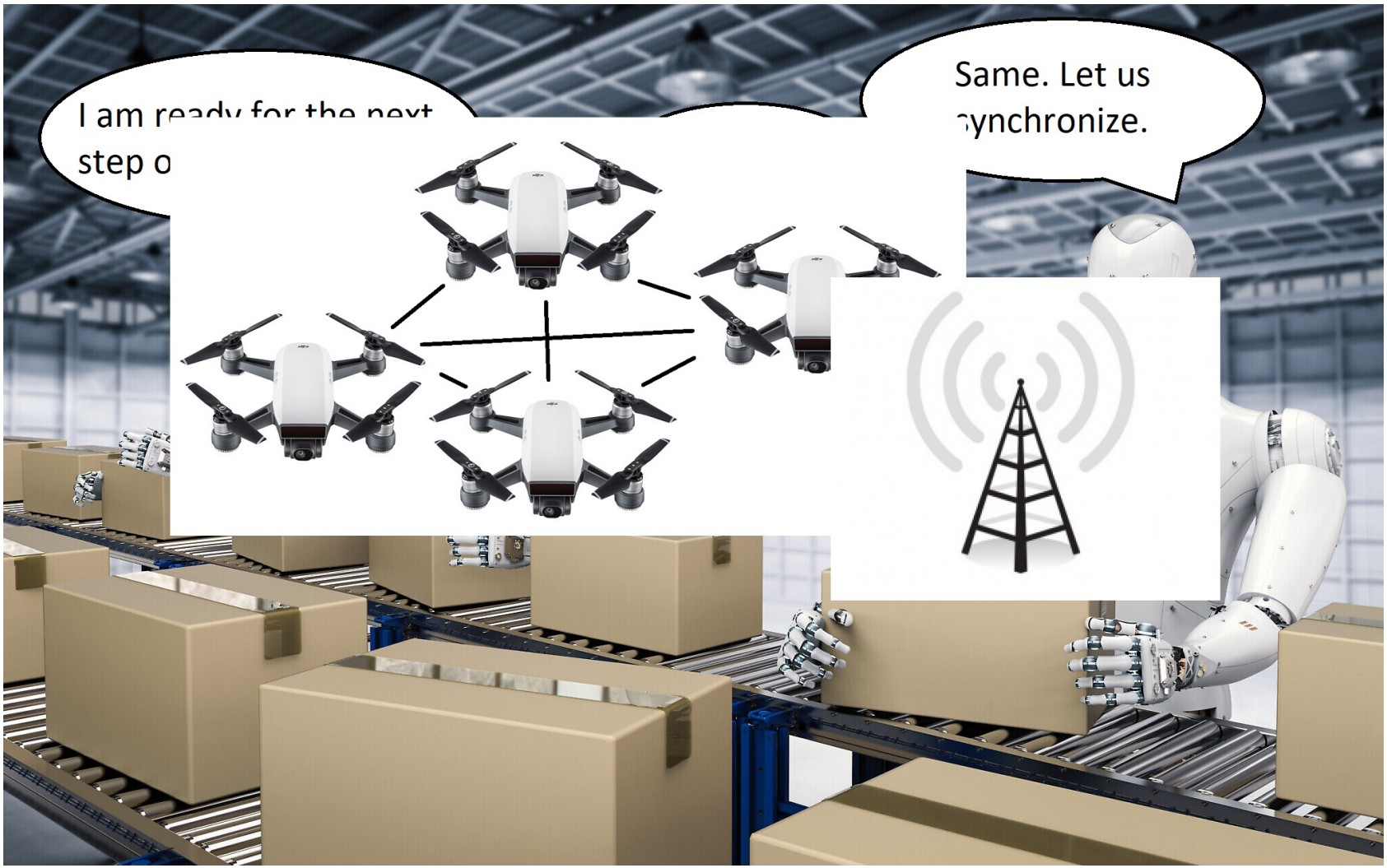
Motivation


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

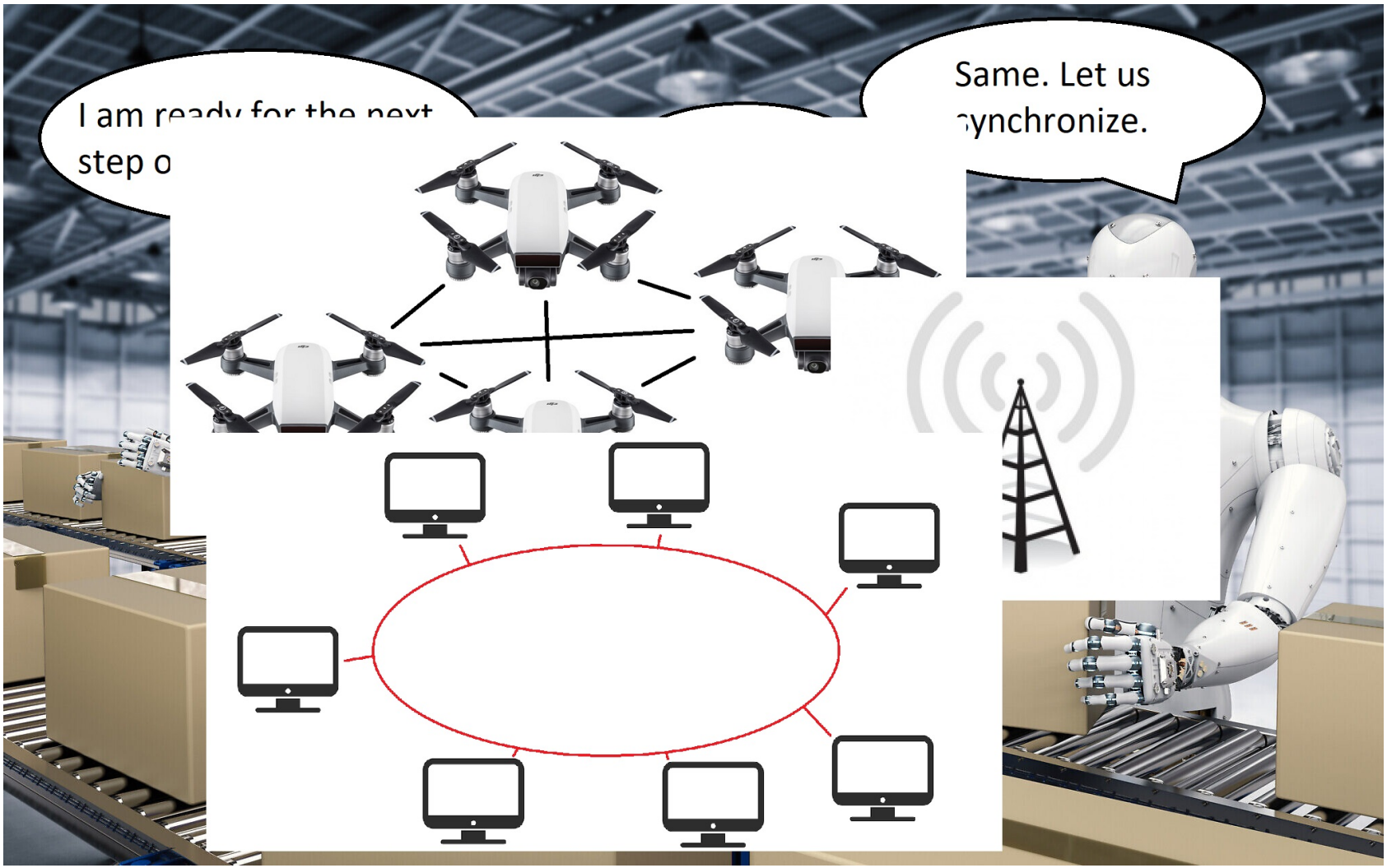
Motivation



Motivation

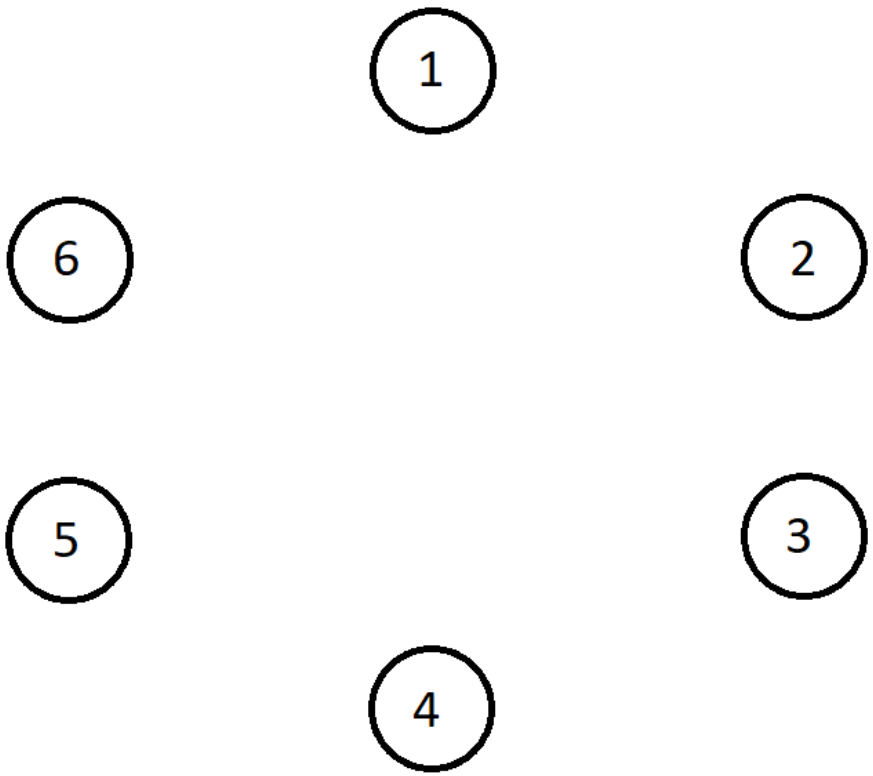


Motivation



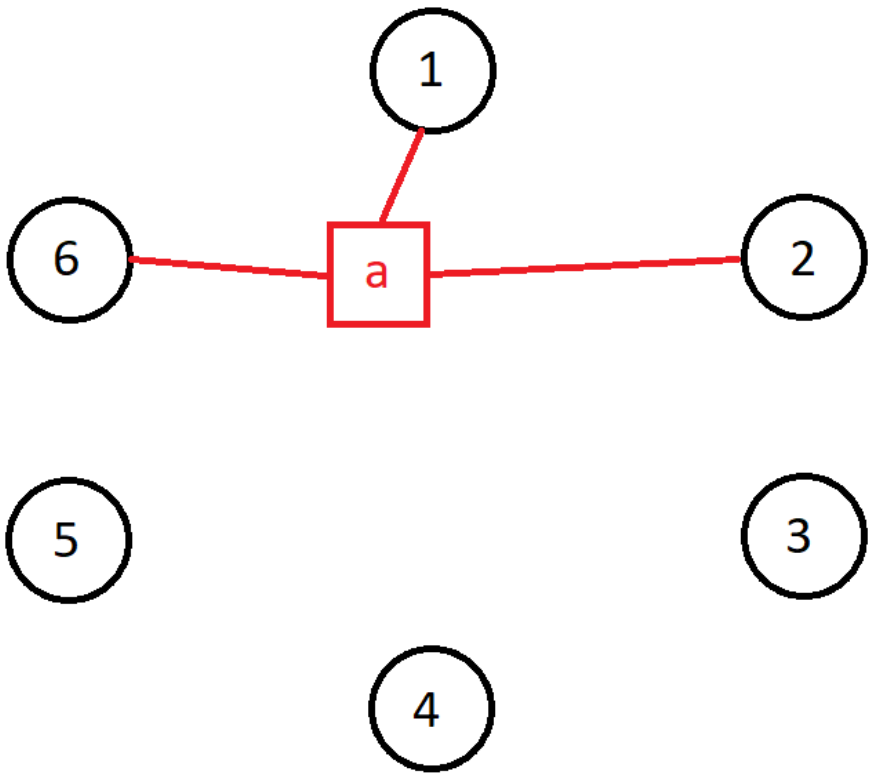
Distributed setting

Independent processes



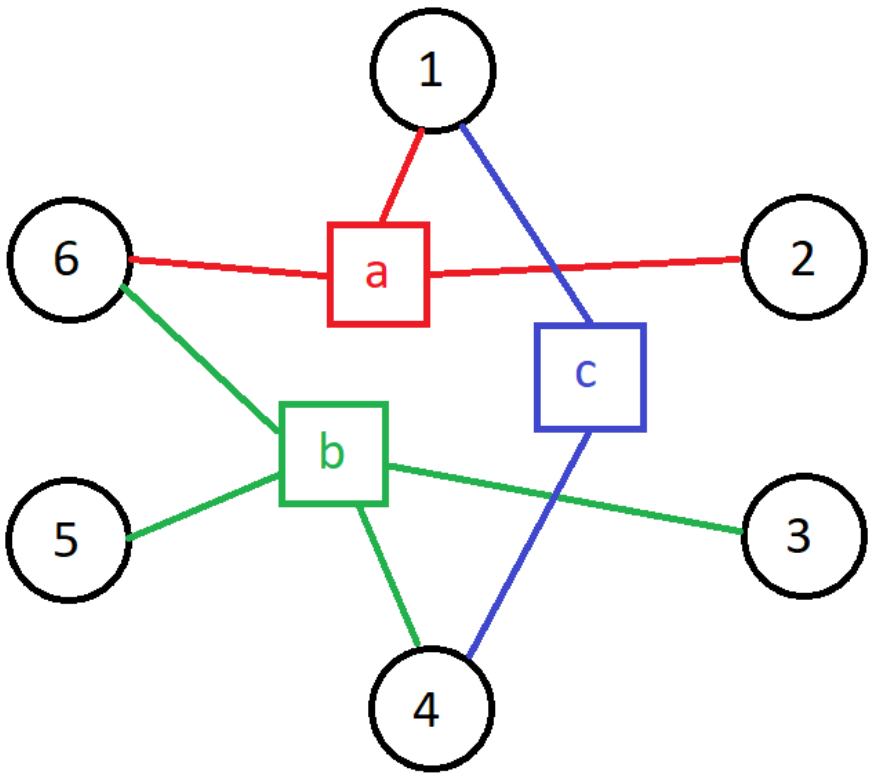
Distributed setting

Independent processes communicating over channels



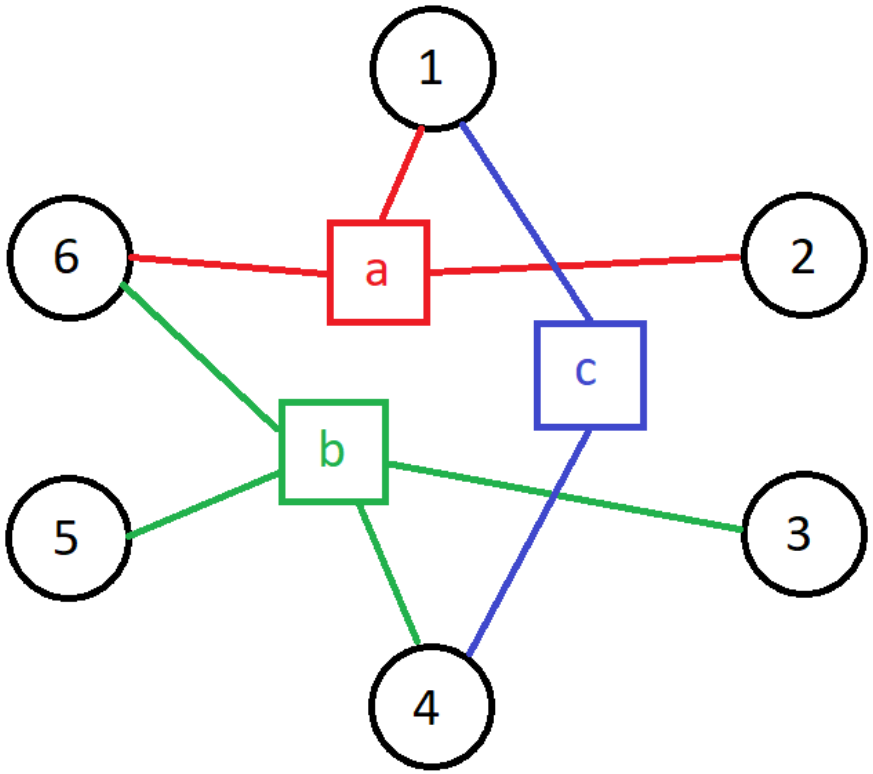
Distributed setting

Independent processes communicating over channels



Distributed setting

Independent processes communicating over channels

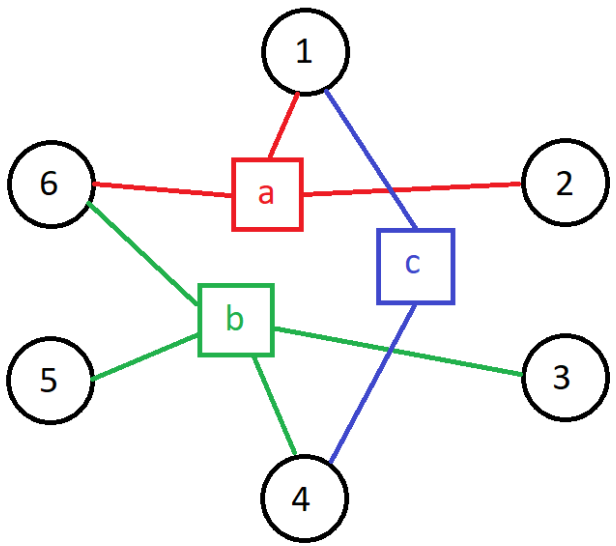


Some assumptions

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

Fixed or reconfigurable communications

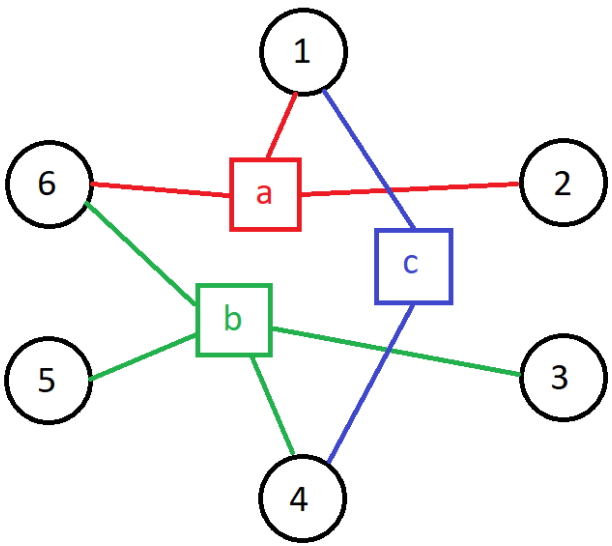
Fixed:



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

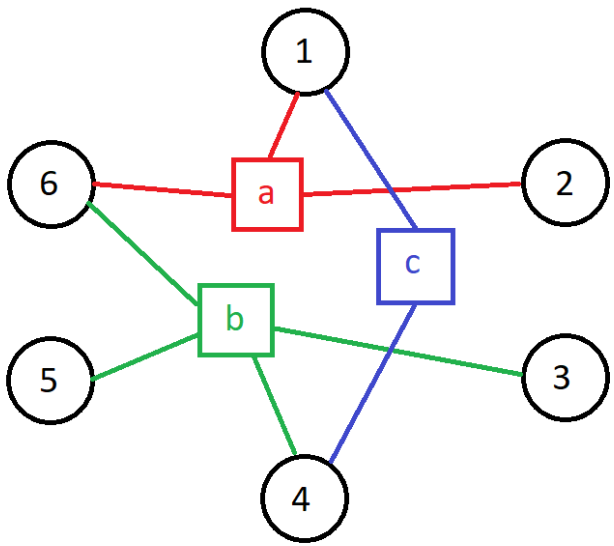
Fixed or reconfigurable communications

Fixed:



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

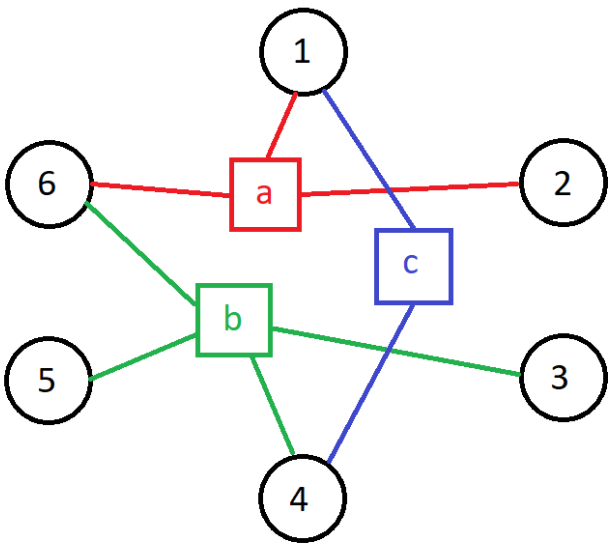
Reconfigurable:



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

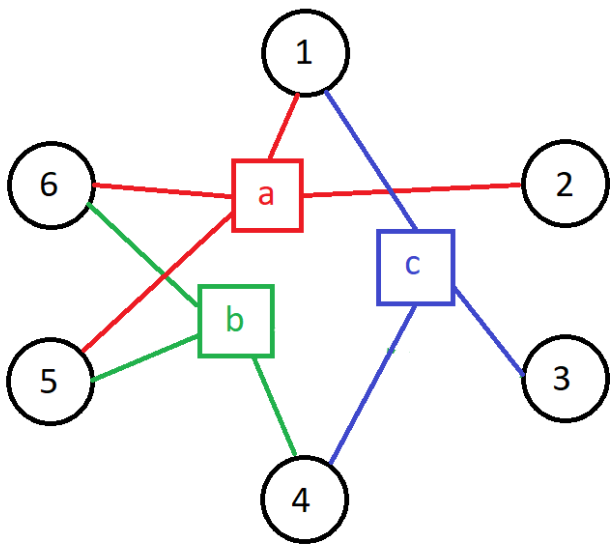
Fixed or reconfigurable communications

Fixed:



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

Reconfigurable:



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

Fixed communication model

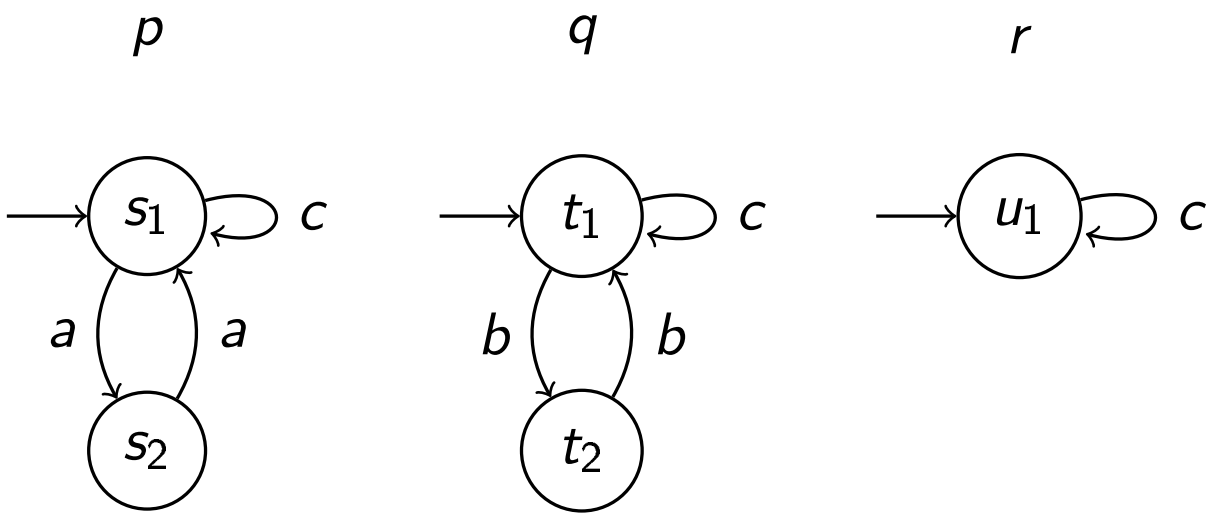
Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

Fixed communication model

Zielonka's Asynchronous Automata (AA)

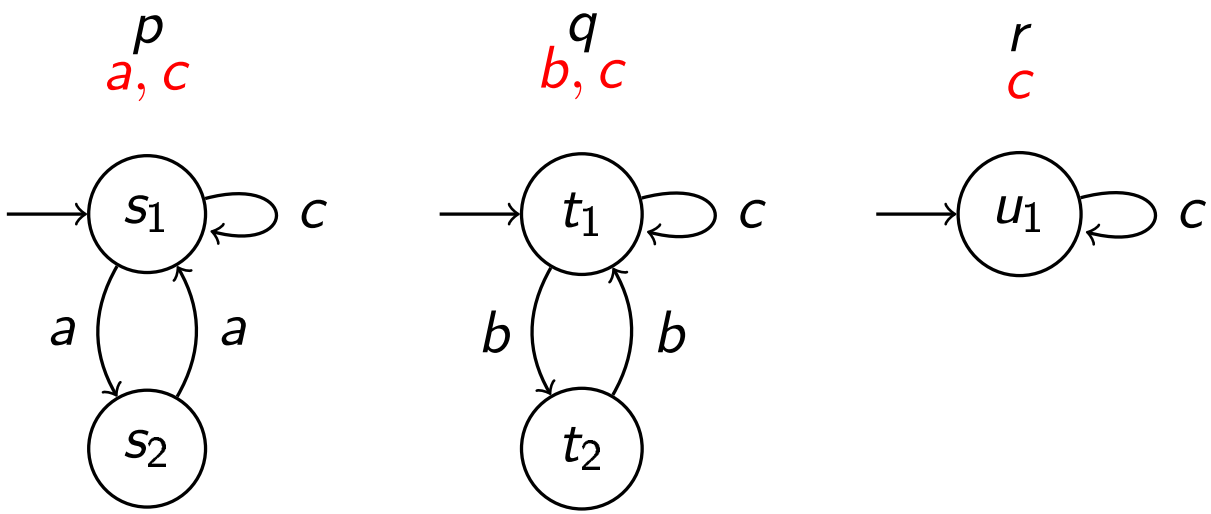
Fix set of processes, channels, and communication structure



Fixed communication model

Zielonka's Asynchronous Automata (AA)

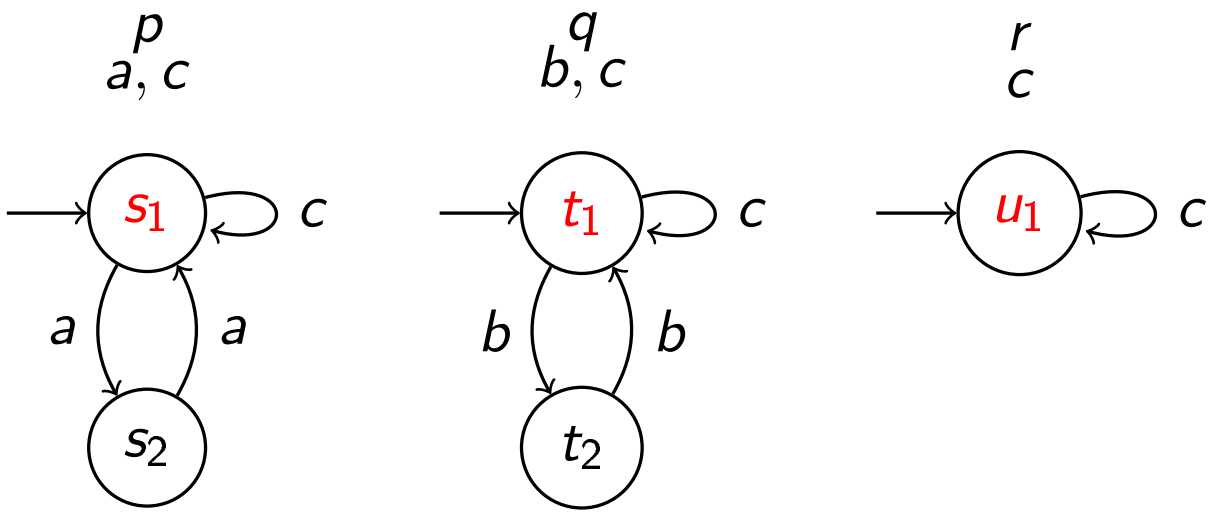
Fix set of processes, channels, and communication structure



Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

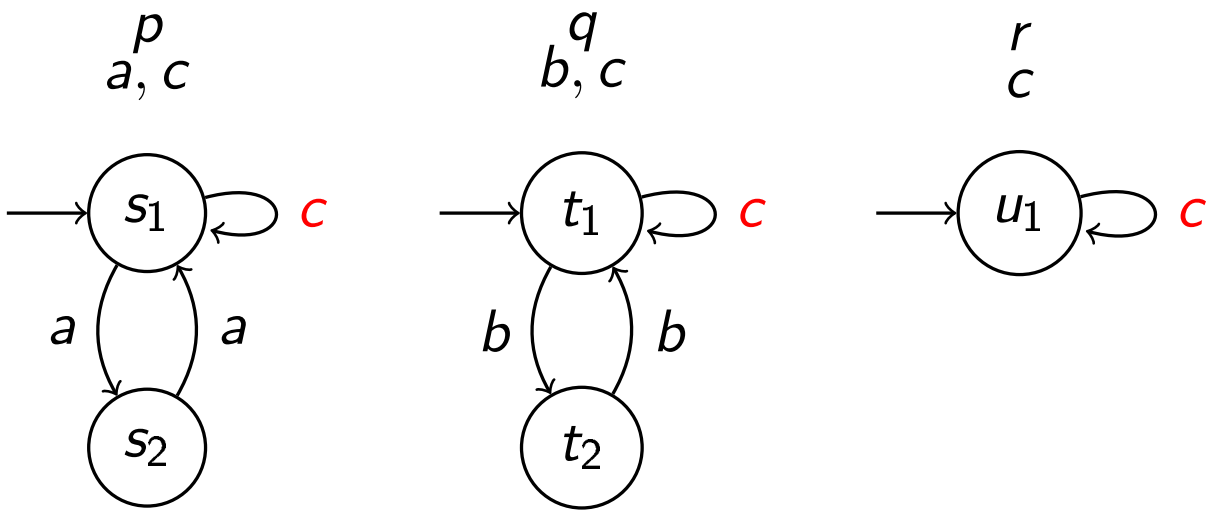


$\rho = (s_1, t_1, u_1)$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

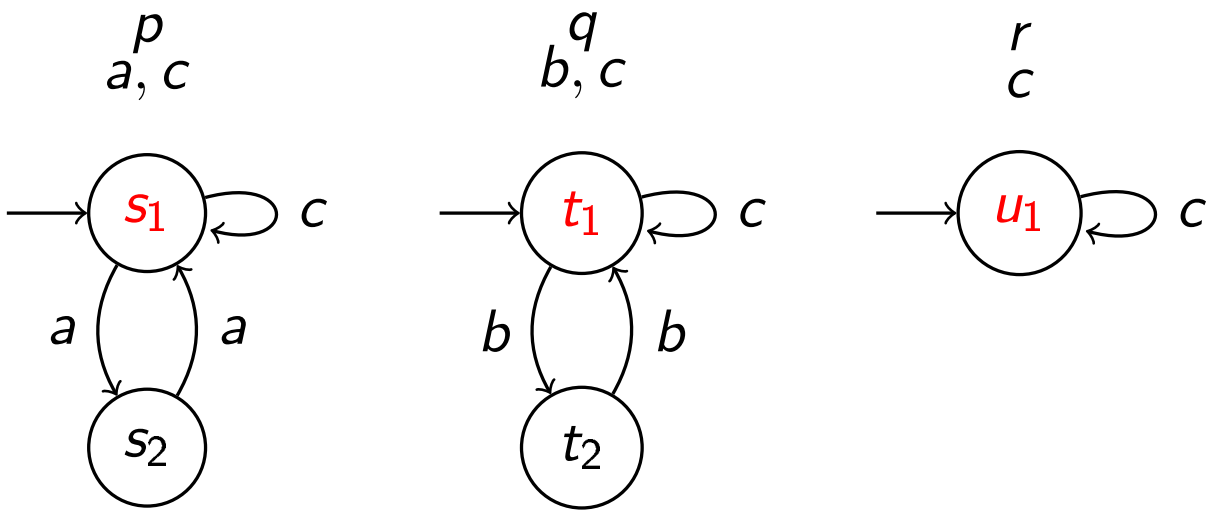


$\rho = (s_1, t_1, u_1) \rightarrow^c$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

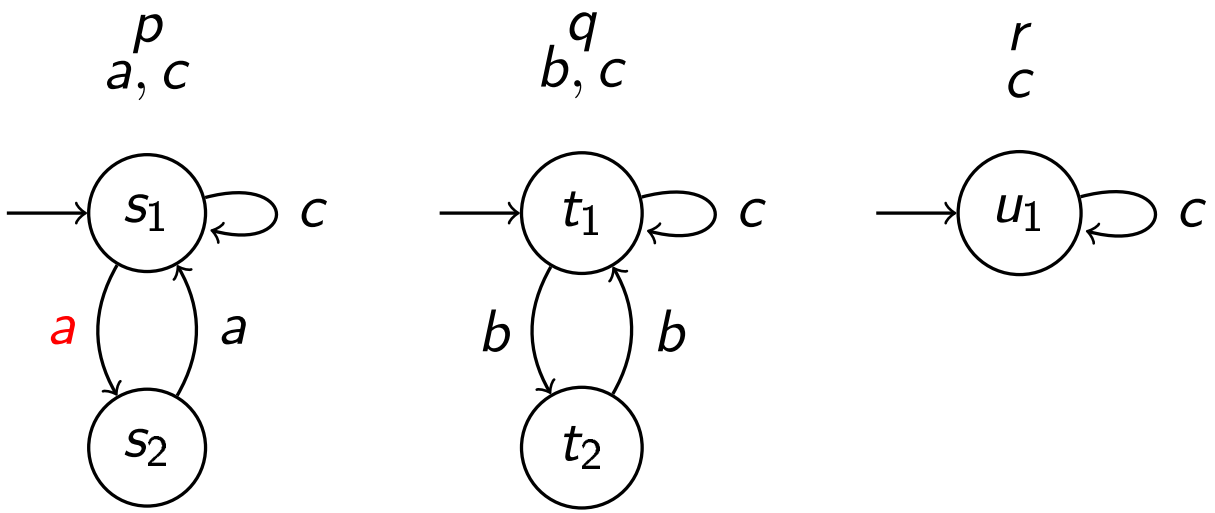


$$\rho = (s_1, t_1, u_1) \rightarrow^c (s_1, t_1, u_1)$$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

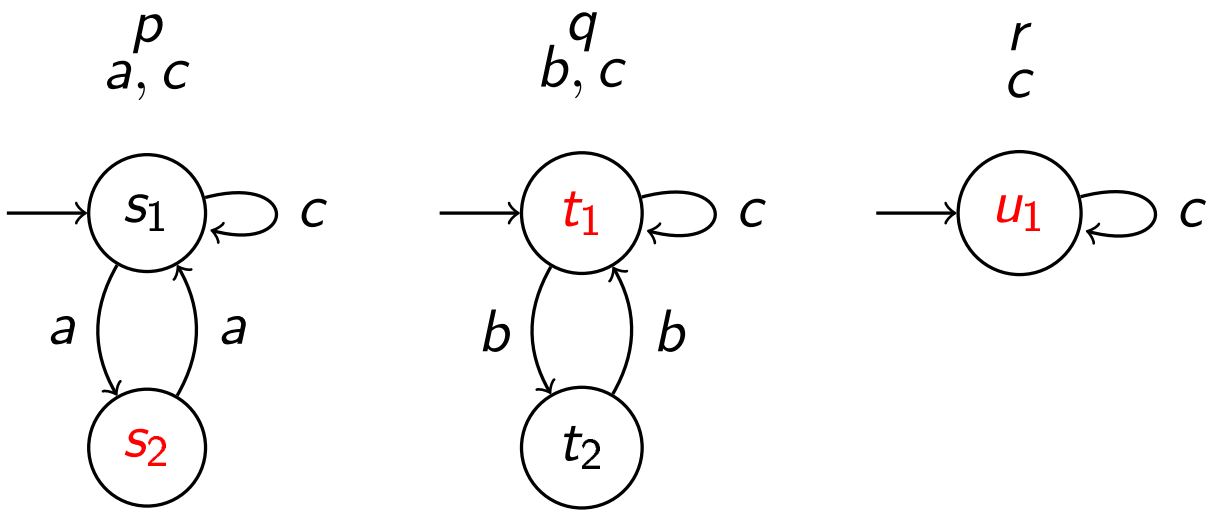


$$\rho = (s_1, t_1, u_1) \rightarrow^c (s_1, t_1, u_1) \rightarrow^a$$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

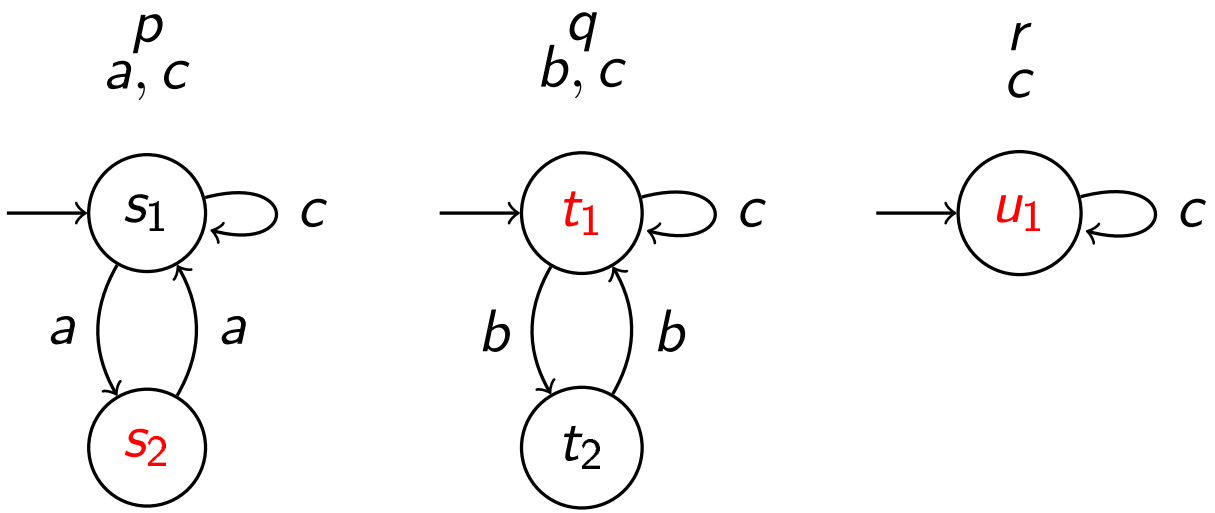


$$\rho = (s_1, t_1, u_1) \rightarrow^c (s_1, t_1, u_1) \rightarrow^a (s_2, t_1, u_1)$$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

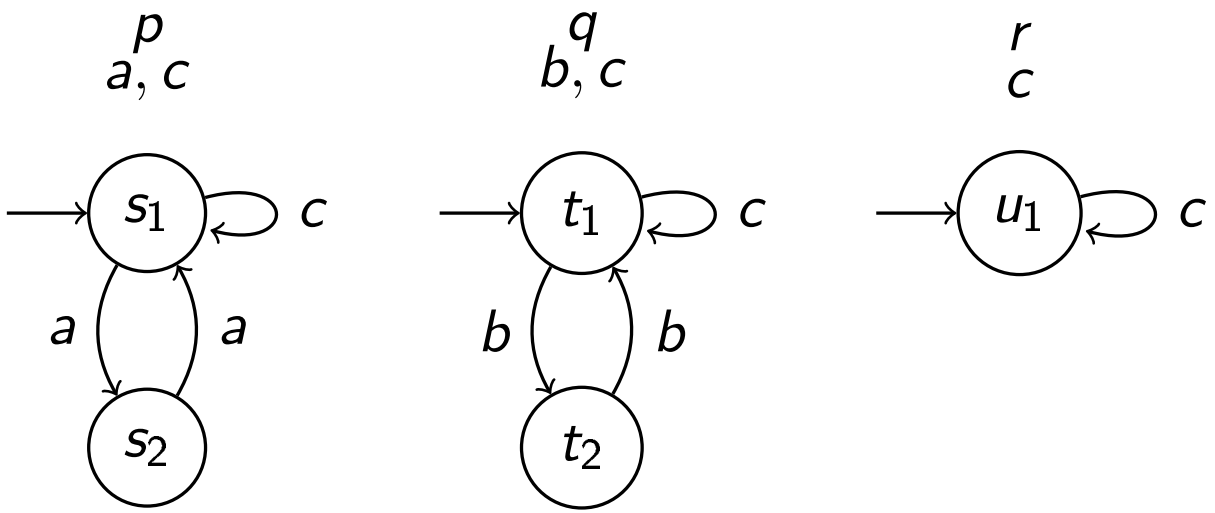


$$\rho = (s_1, t_1, u_1) \to^{c} (s_1, t_1, u_1) \to^{a} (s_2, t_1, u_1) \not\to^{c}$$

Fixed communication model

Zielonka's Asynchronous Automata (AA)

Fix set of processes, channels, and communication structure

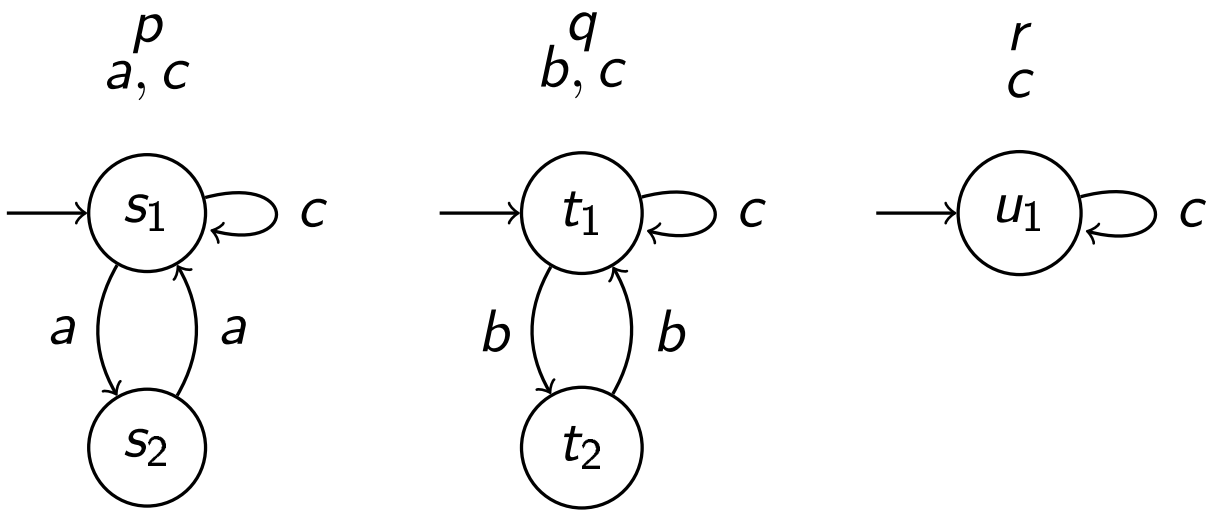


$\rho = (s_1, t_1, u_1) \to^c (s_1, t_1, u_1) \to^a (s_2, t_1, u_1) \to^b \ldots$
$w = cab \ldots$

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

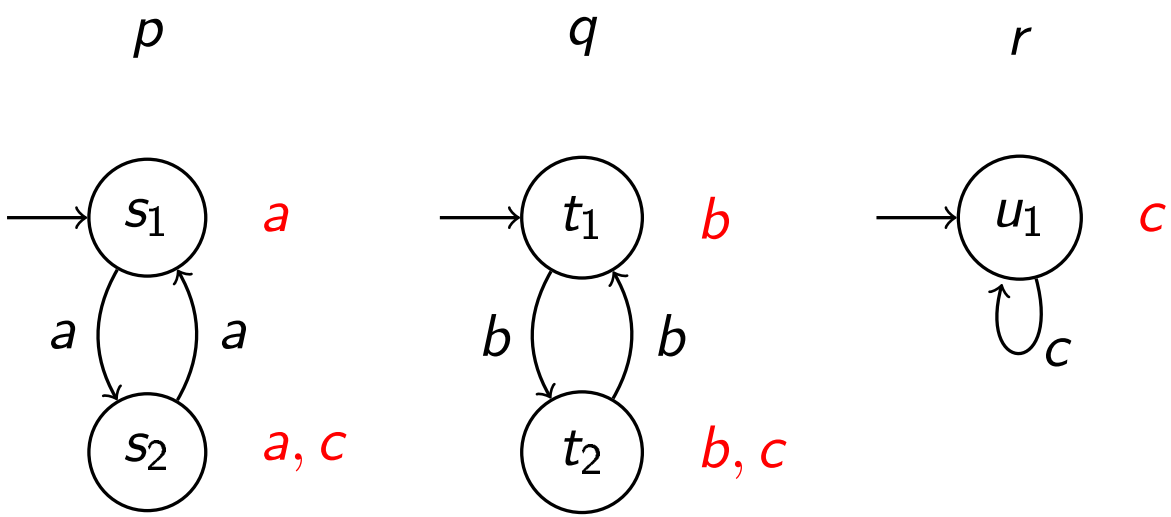
Channeled Transition Systems (CTS)

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

Reconfigurable model

Channeled Transition Systems (CTS)

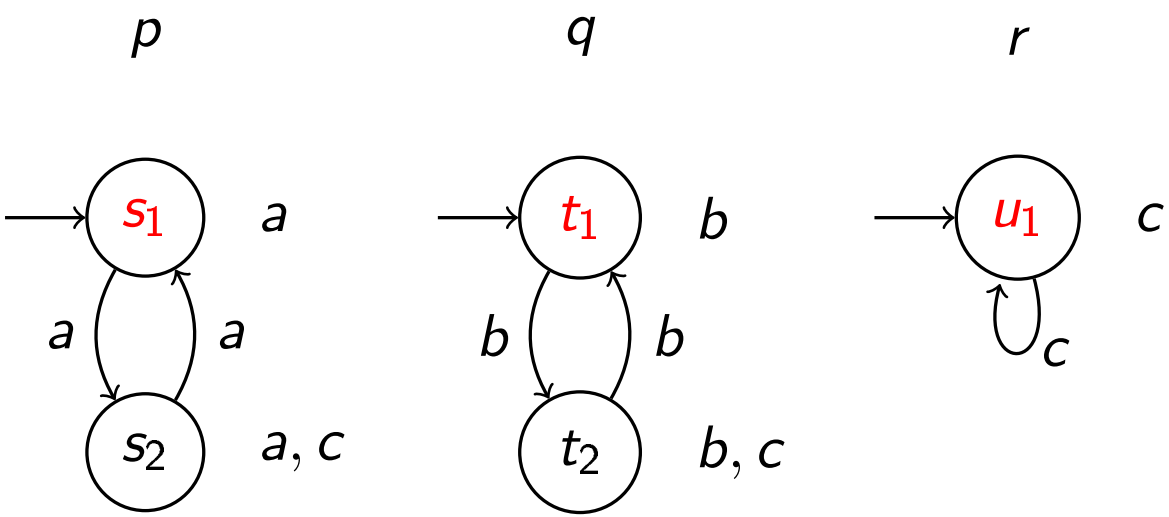
Fix set of processes, channels, and ~~communication structure~~



Reconfigurable model

Channeled Transition Systems (CTS)

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

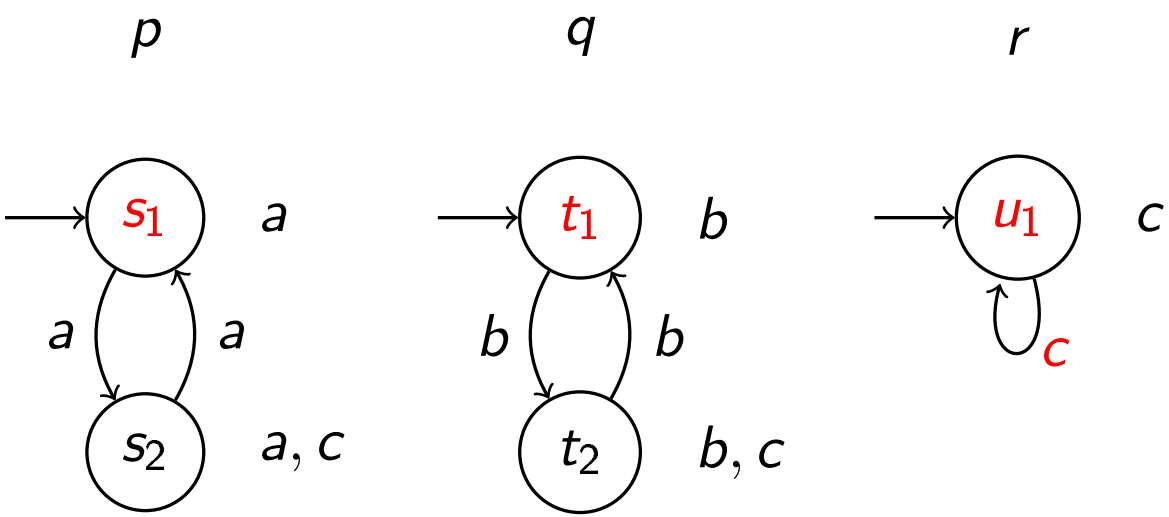


$\rho = (s_1, t_1, u_1)$

Reconfigurable model

Channeled Transition Systems (CTS)

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

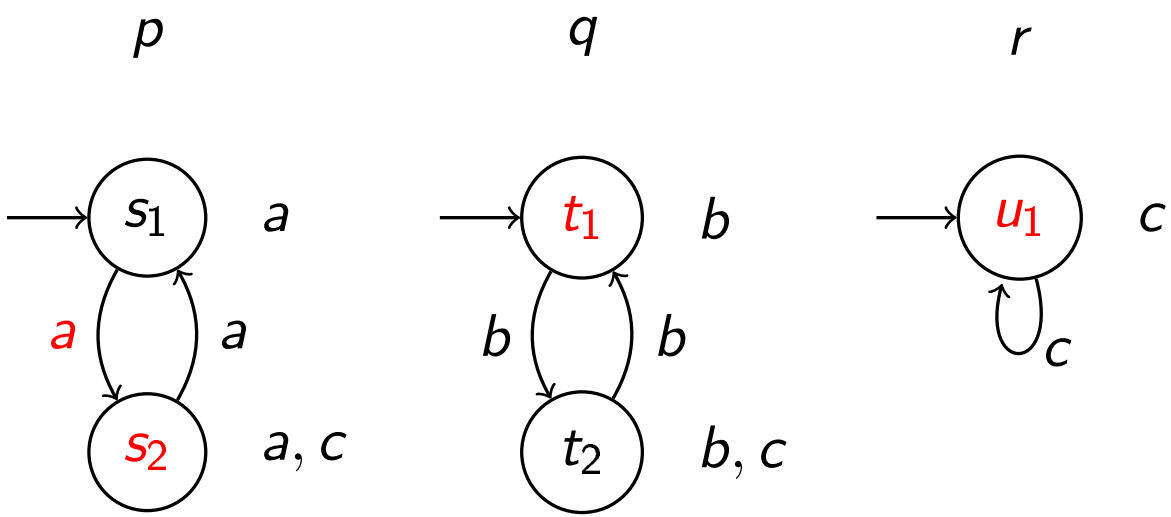


$$\rho = (s_1, t_1, u_1) \rightarrow^c (s_1, t_1, u_1)$$

Reconfigurable model

Channeled Transition Systems (CTS)

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

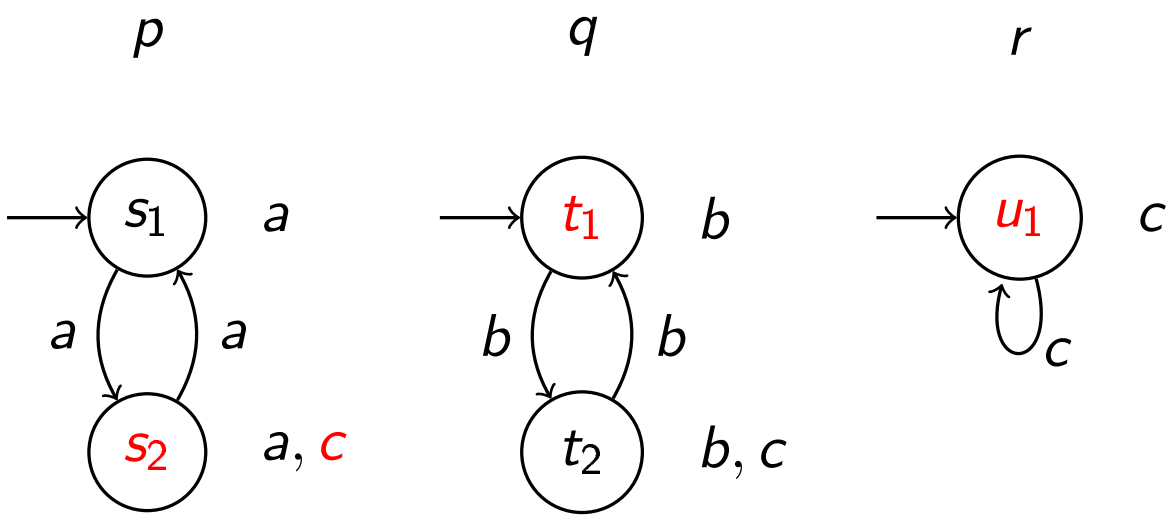


$$\rho = (s_1, t_1, u_1) \rightarrow^c (s_1, t_1, u_1) \rightarrow^a (s_2, t_1, u_1)$$

Reconfigurable model

Channeled Transition Systems (CTS)

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



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

Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

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

Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

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

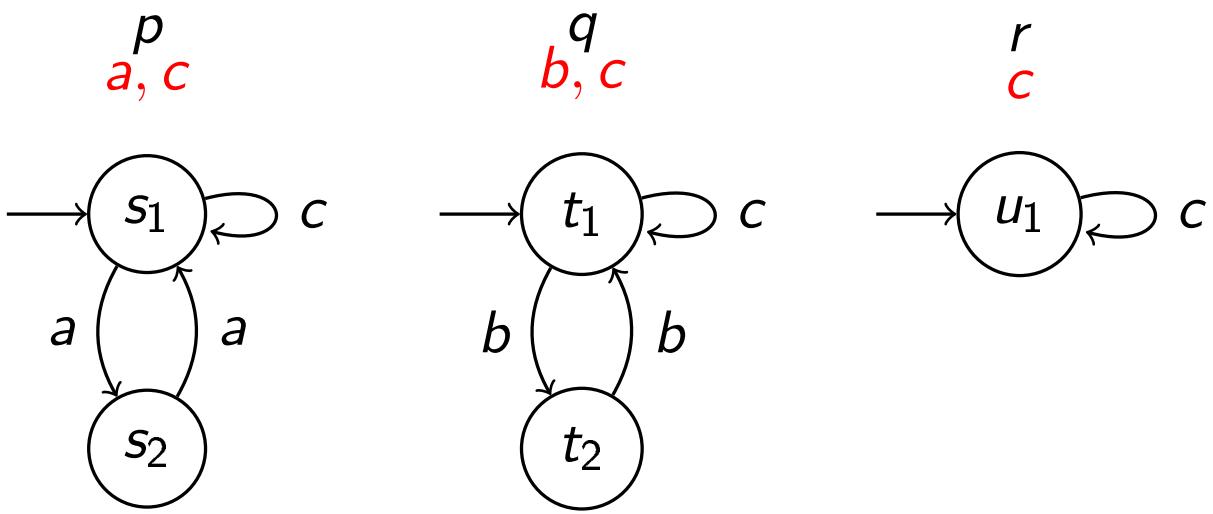
Proof idea: Always listen to the (fixed) set of channels, whatever the state

Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

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

Proof idea: Always listen to the (fixed) set of channels, whatever the state

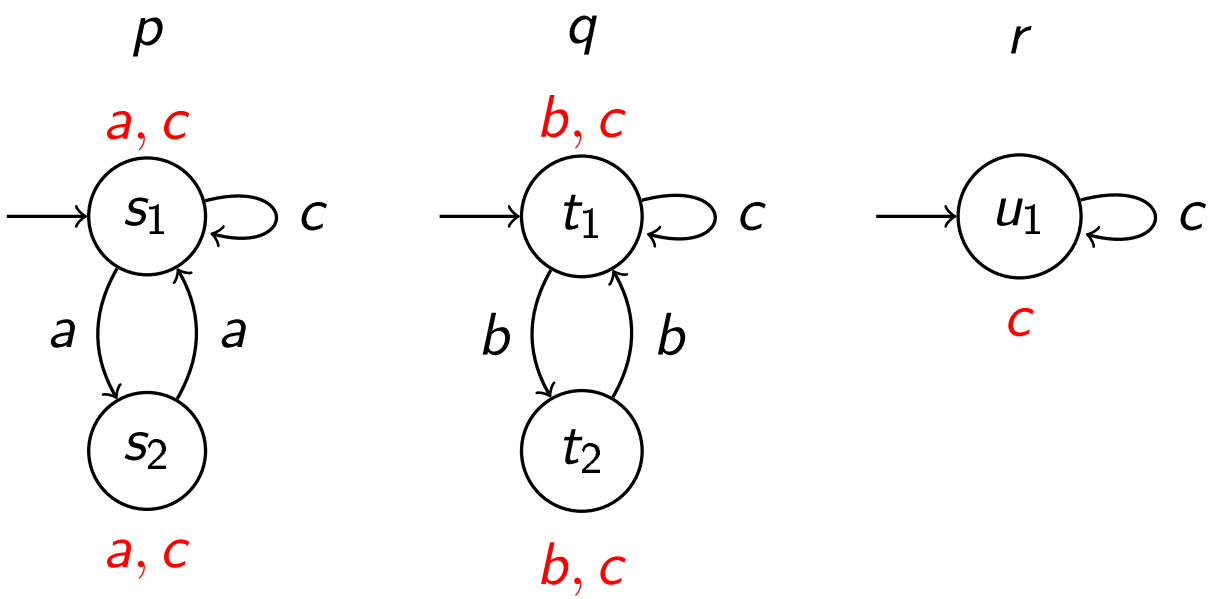


Fixed vs. Reconfigurable

Result 1: Fixed → Reconfigurable

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

Proof idea: Always listen to the (fixed) set of channels, whatever the state



Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

Fixed vs. Reconfigurable

Result 2: Reconfigurable → Fixed

Same with other direction!

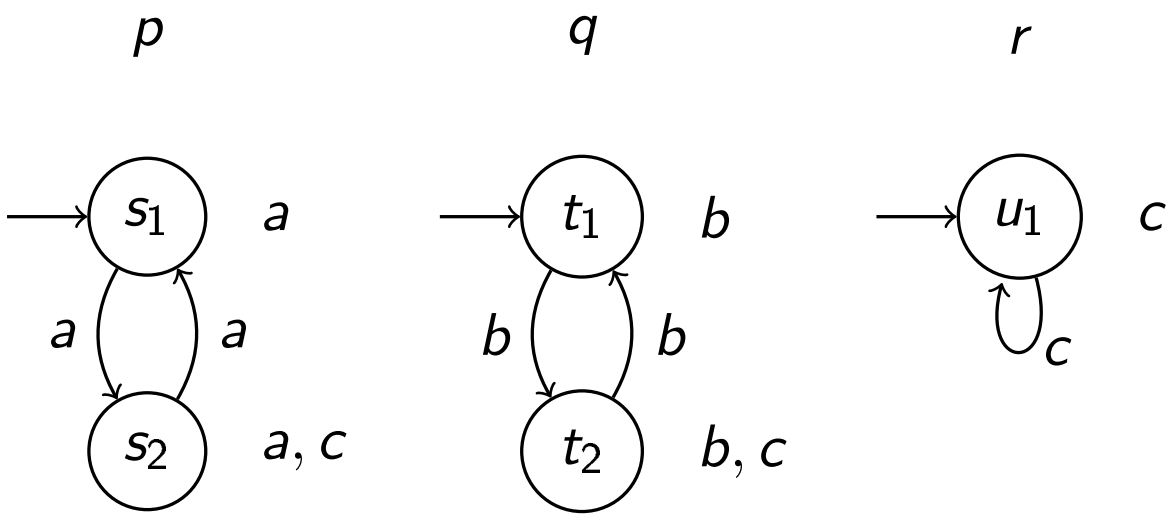
Proof idea: Listen to every channel, ignore (but don't block!) communications you were not supposed to listen to originally

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

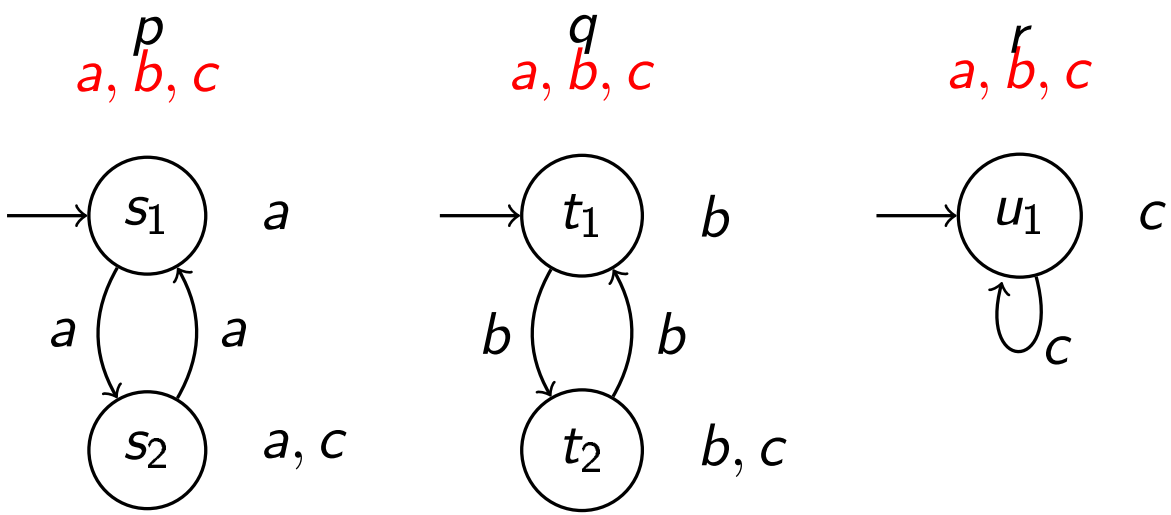


Fixed vs. Reconfigurable

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

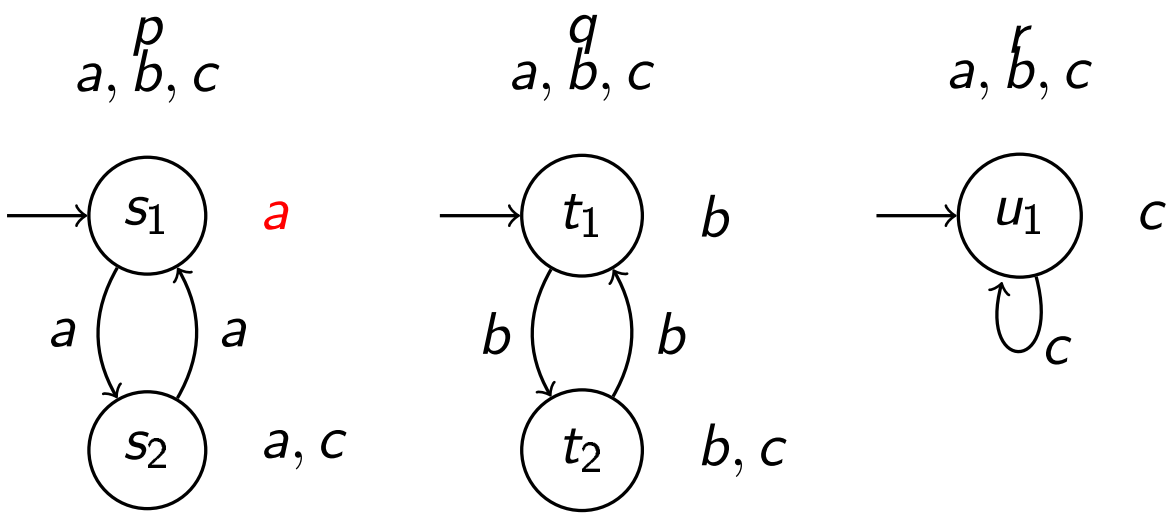


Fixed vs. Reconfigurable

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

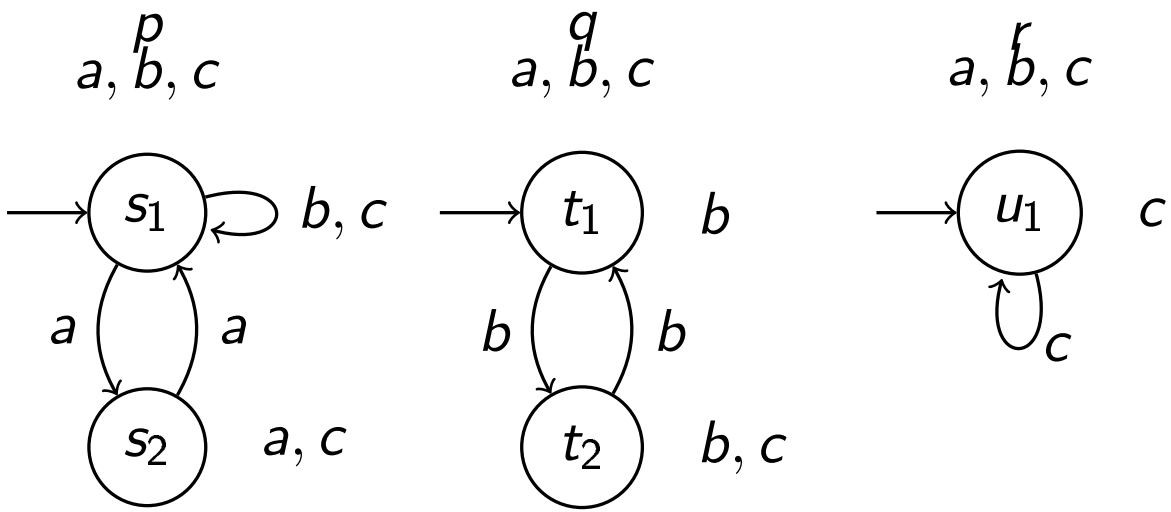


Fixed vs. Reconfigurable

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

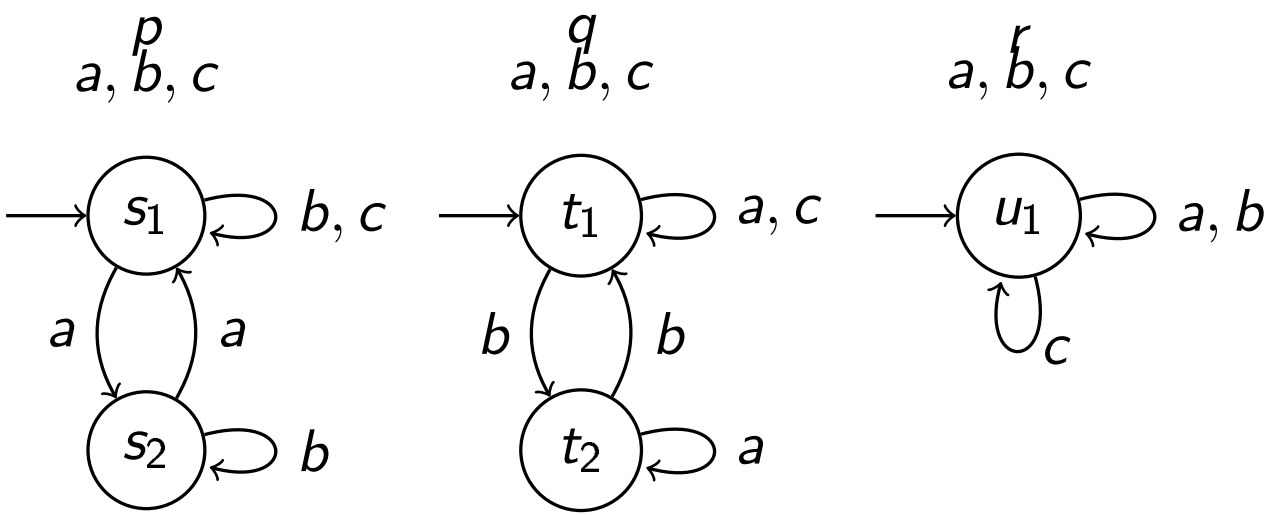


Fixed vs. Reconfigurable

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?

Result 3: No better general construction

There is a CTS with no "nice" equivalent AA

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

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
- 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
- How to measure (then minimize) the amount of communications in a system while keeping the same behaviors?

Thank you for your attention!