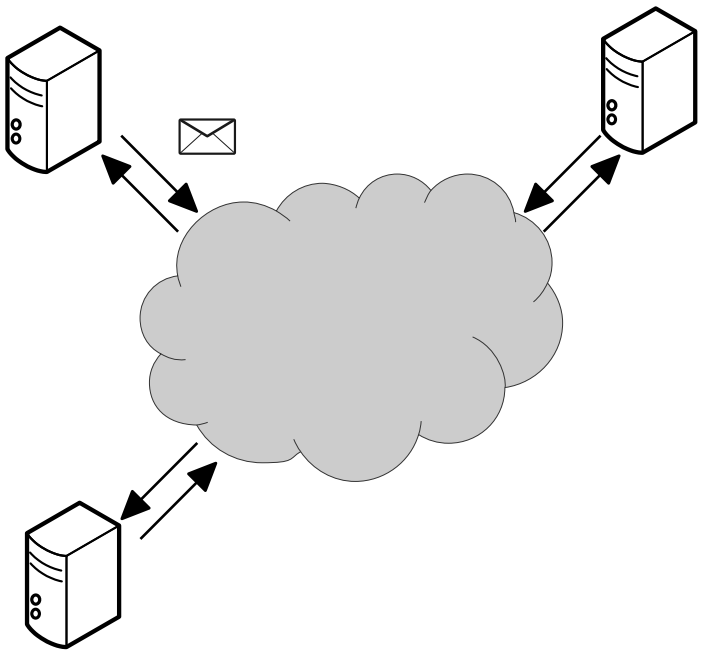
Verification of message-passing programs: model checking and session types perspectives

Damien Zufferey
2020.10.7

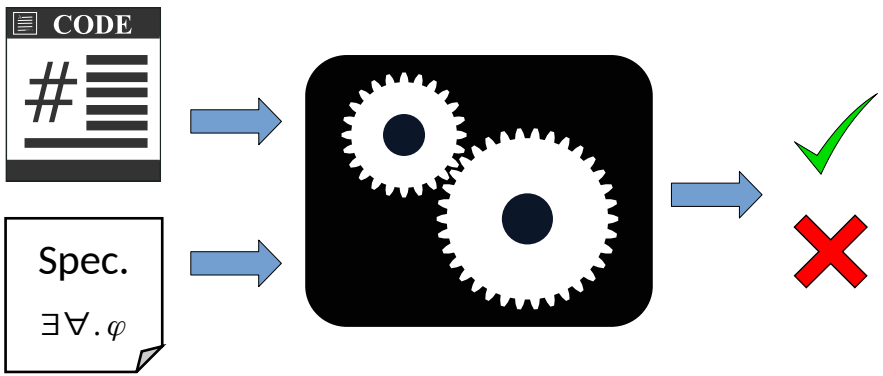
What is this about ?

Distributed systems
(message-passing)

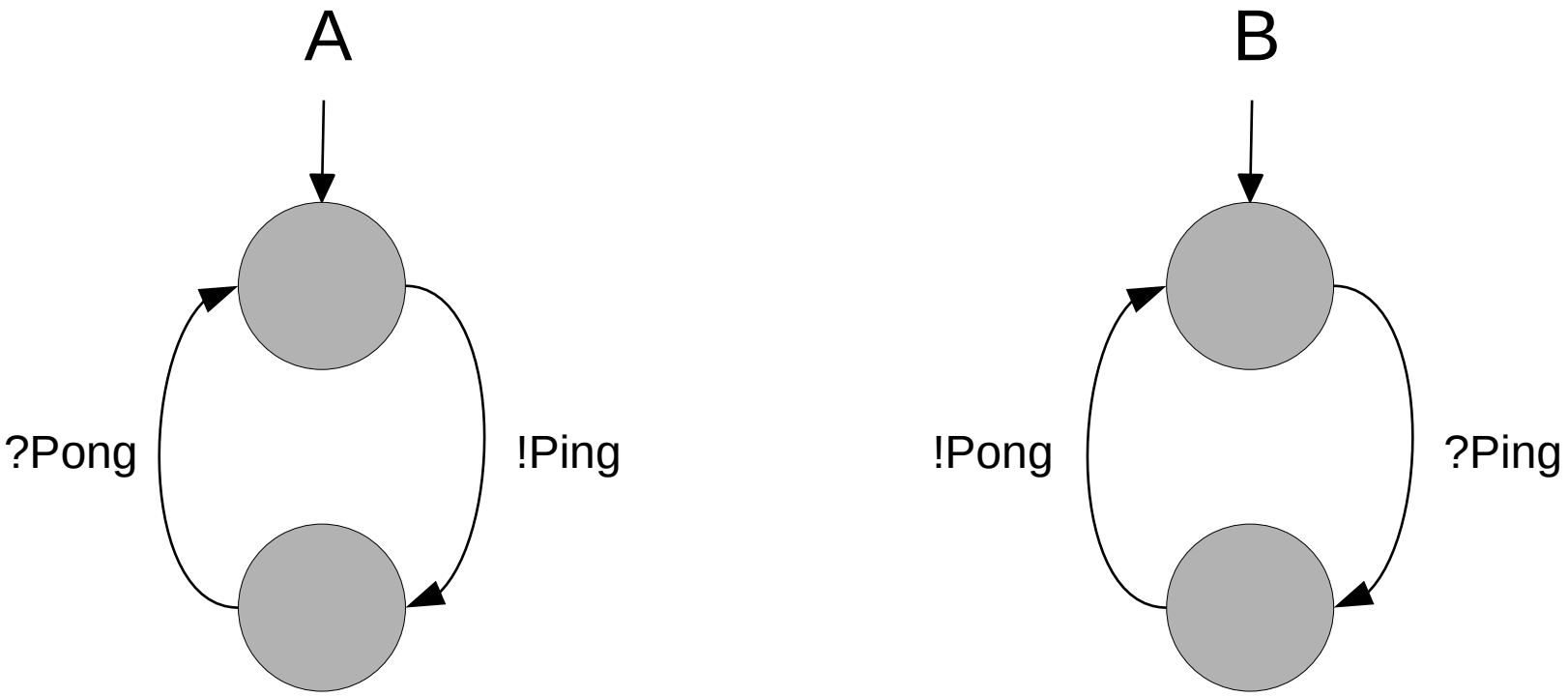


+

Verification

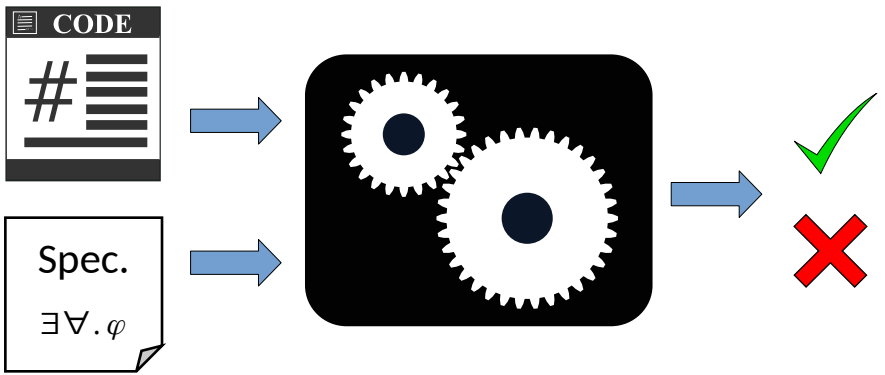


“Hello world” of message-passing



CSP notation: ‘!’ is send, ‘?’ is receive

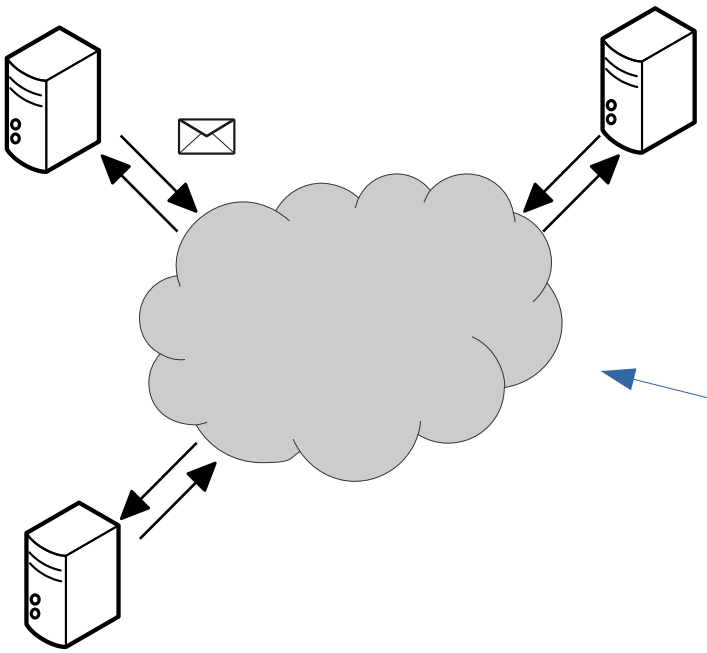
What to verify?



A communication protocol is correctly implemented:

- messages follow a prescribed order
- deadlock-free
- no dangling messages

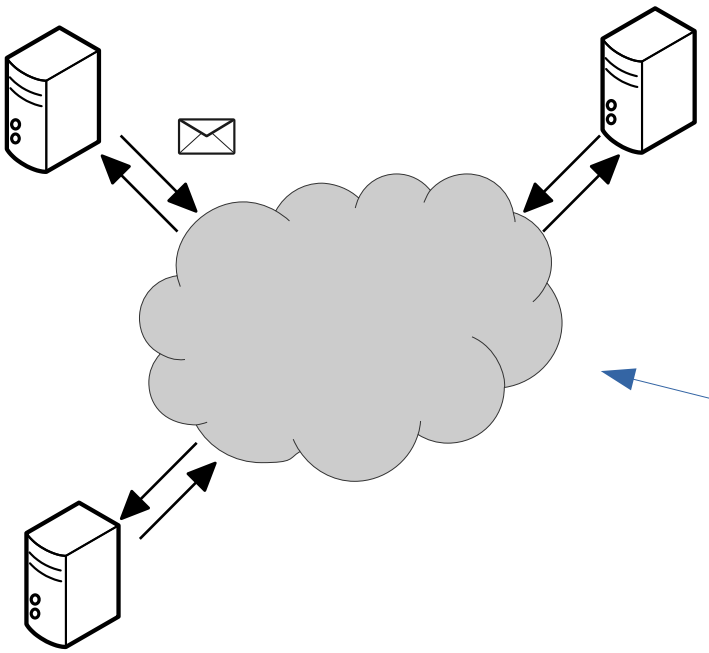
Model for the software



Programs:
- Finite state machines
- Pushdown automata
- Turing machines

Communication channels:
- routing: point-to-point, broadcast, ...
- capacity: unbounded, bounded
- reliability: reliable, lossy, fair, ...
- ordering: FIFO, bag

Communicating state machines (CSM)



Programs:
- **Finite state machines**
- Pushdown automata
- Turing machines

Communication channels:
- routing: **point-to-point**, broadcast, ...
- capacity: **unbounded**, bounded
- reliability: **reliable**, lossy, fair, ...
- ordering: **FIFO**, bag

CSM: expressive power

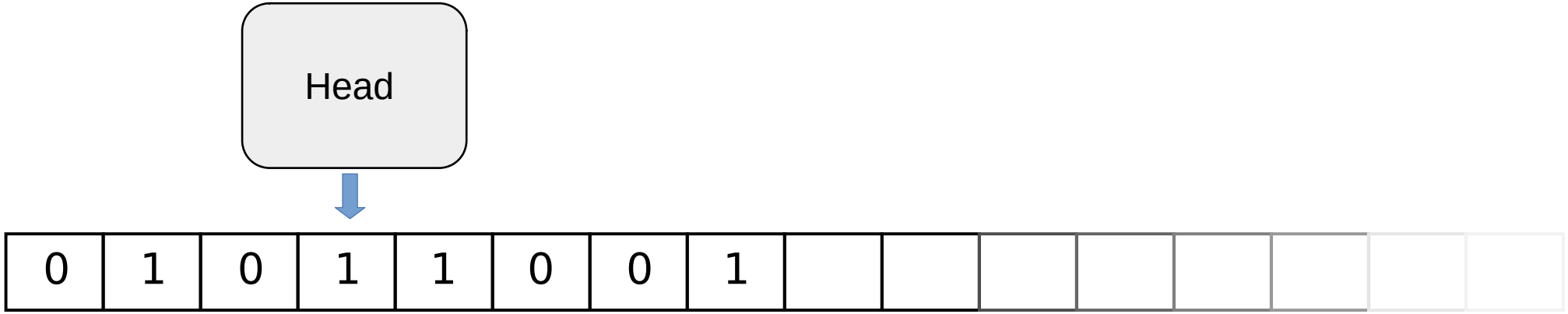
- Knowing the expressive power tells us what method can or cannot be applied.

  (Don’t try to solve undecidable problems)

- It can tell us what feature of the model is important.
- Unfortunately, CSM can simulate Turing machines.

Turing machine (TM)

- TM = finite control + unbounded tape (RAM)
- Operations: read/write memory, move left/right



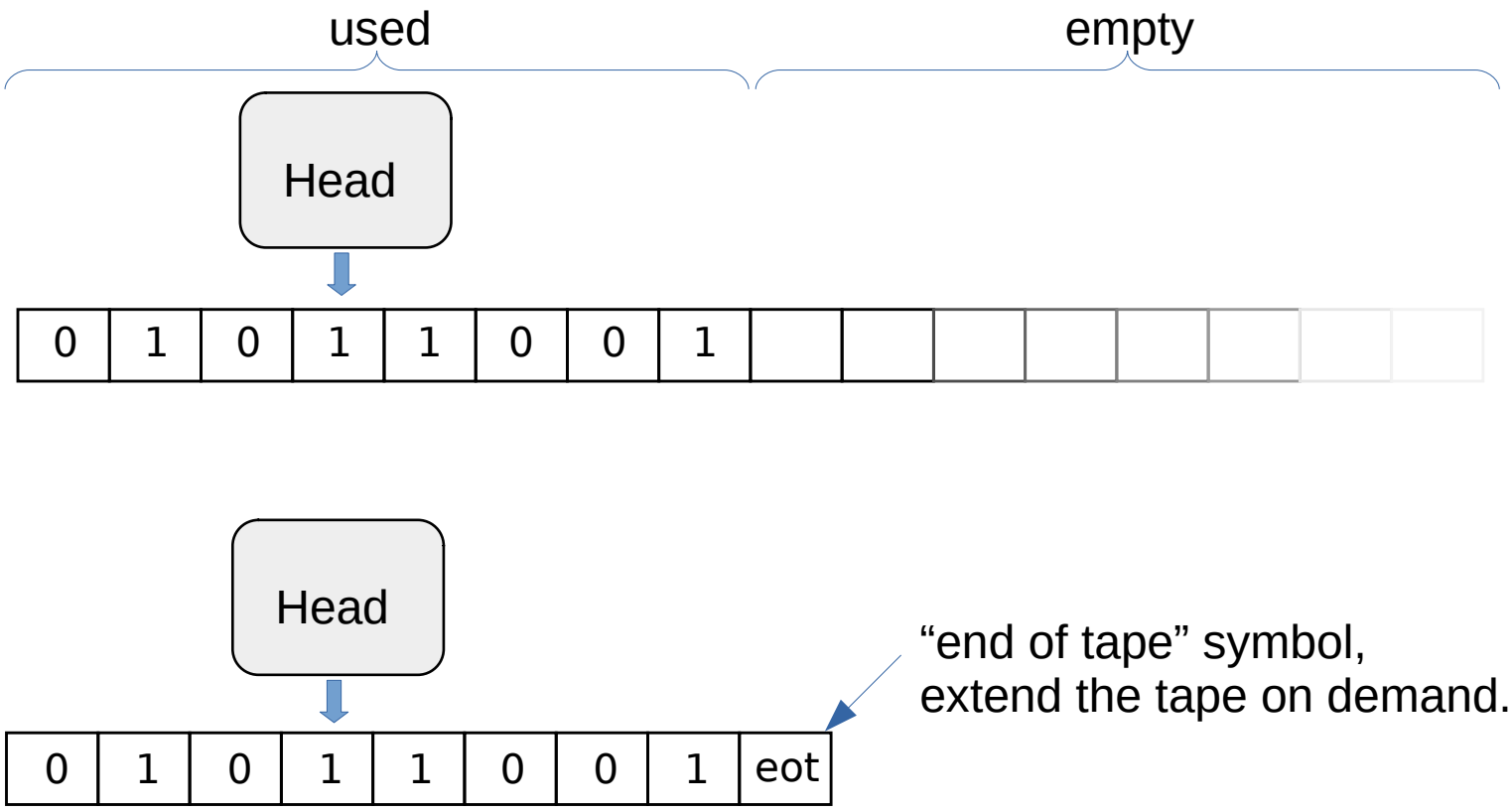
TM → CSM

Idea:

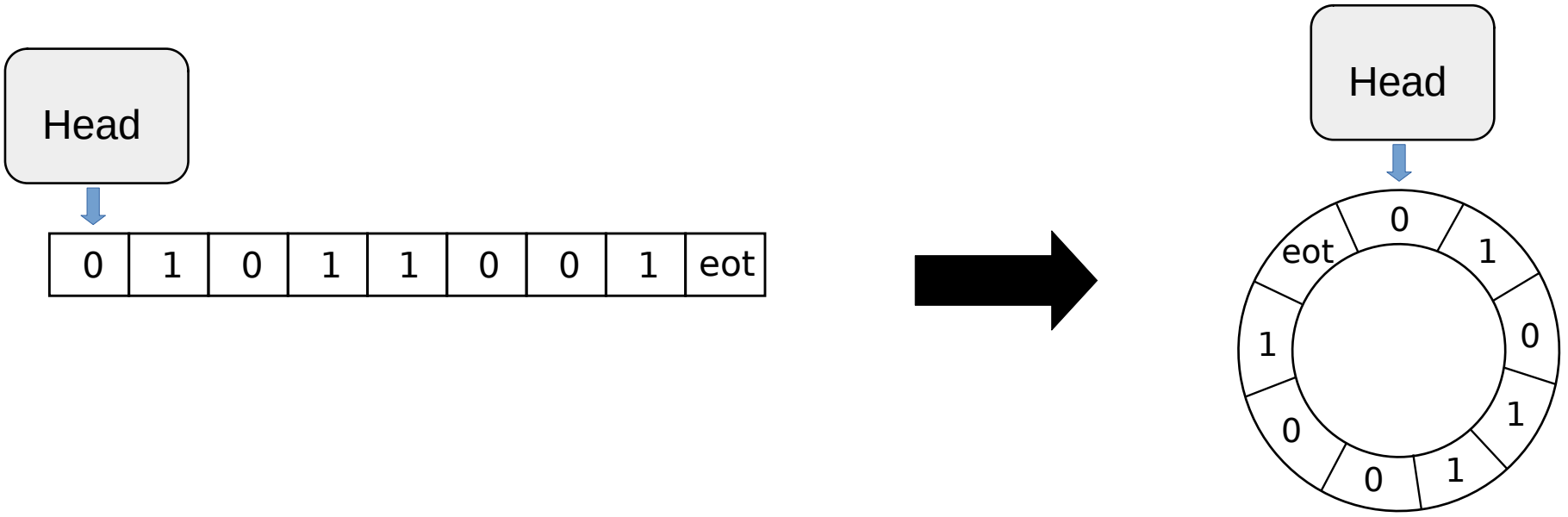
use the communication channels to store the content of the tape.

1) Store only the (finite) written part
2) Turn the tape into a loop
3) Extract the “Head part” of the tape
4) Split the loop in two parts (2 communication channels)

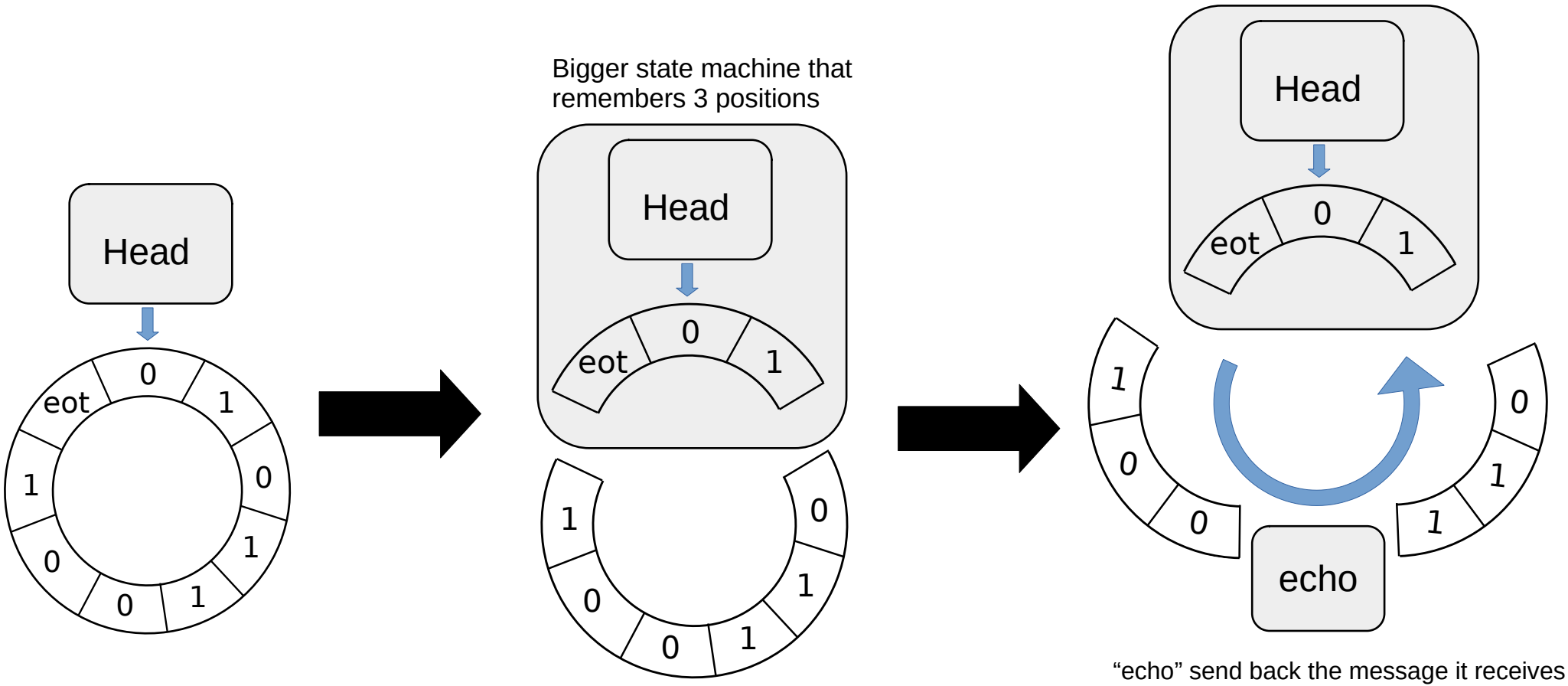
TM → CSM


used
empty
Head
0 1 0 1 1 0 0 1
Head
0 1 0 1 1 0 0 1 eot
“end of tape” symbol,
extend the tape on demand.

TM → CSM

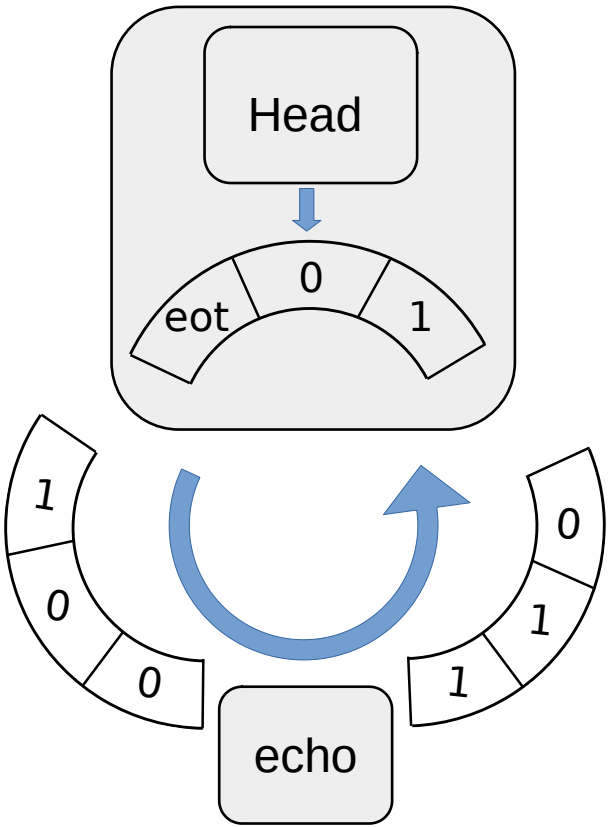


TM → CSM



“echo” send back the message it receives

TM → CSM



Emulating TM operations:

1) read/write: change local state of the head machine

2) move right: send to echo, receive from echo

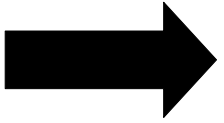
3) move left:
   - insert position marker before current position
   - loop around the tape (move right) until the marker

Other channel models

- capacity: unbounded, **bounded**
- reliability: reliable, lossy, ...
- ordering: FIFO, bag

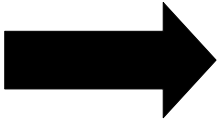
Finite state machine

- capacity: **unbounded**, bounded
- reliability: reliable, lossy, ...
- ordering: FIFO, **bag**



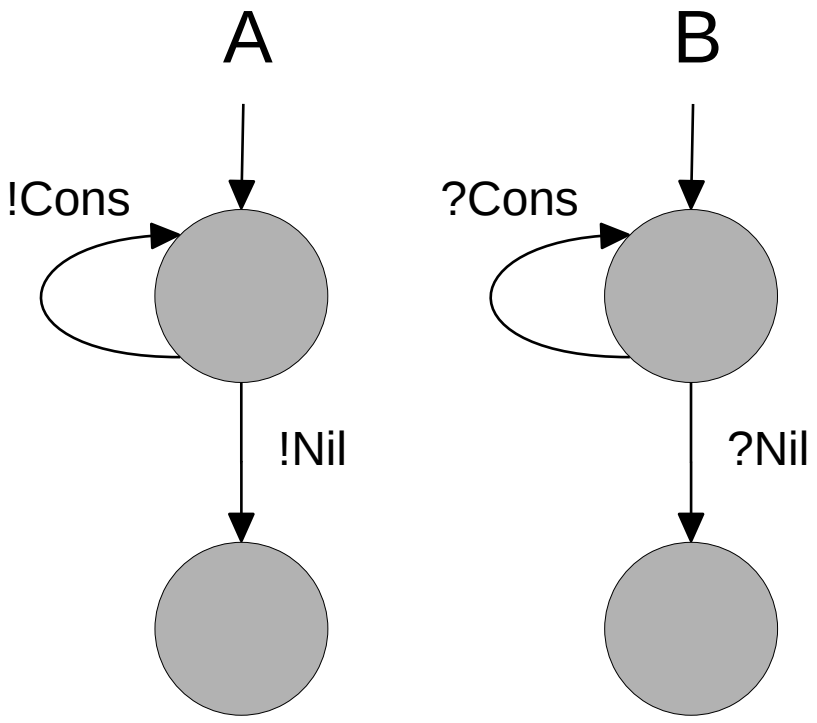
Petri Net

- capacity: **unbounded**, bounded
- reliability: reliable, **lossy**, ...
- ordering: **FIFO**, bag



WSTS

Memory vs communication slack



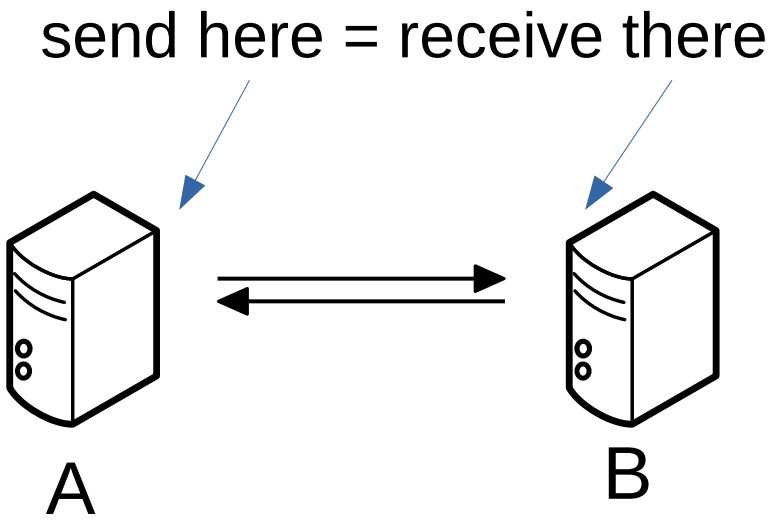
Some protocols using unbounded channels can be harmless.

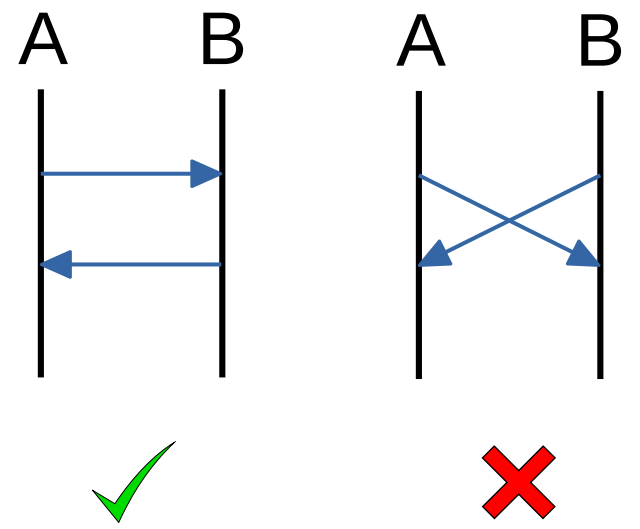
What is the difference between

- channel as memory
  (store information for later)
- communication slack
  (delay in the propagation of information)

Looking at the channels

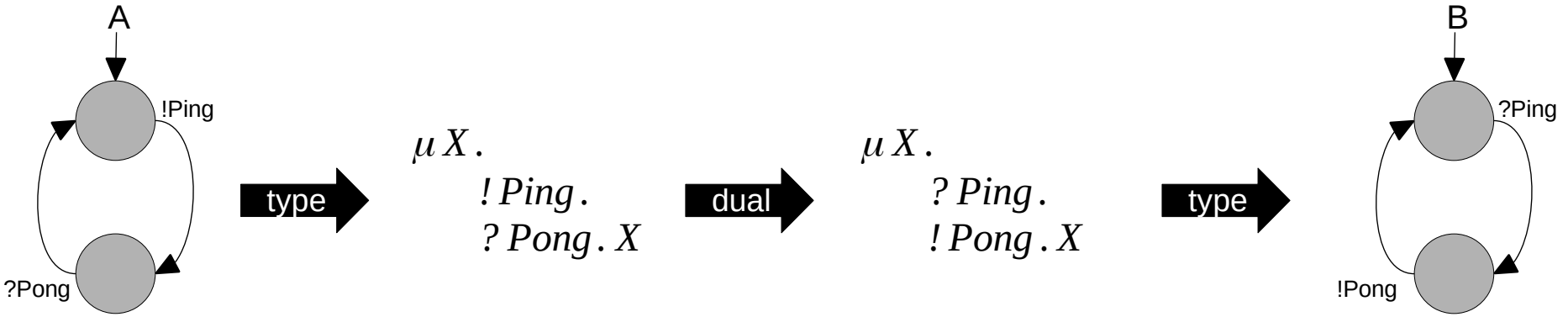
Operations on channel endpoints are **dual**.



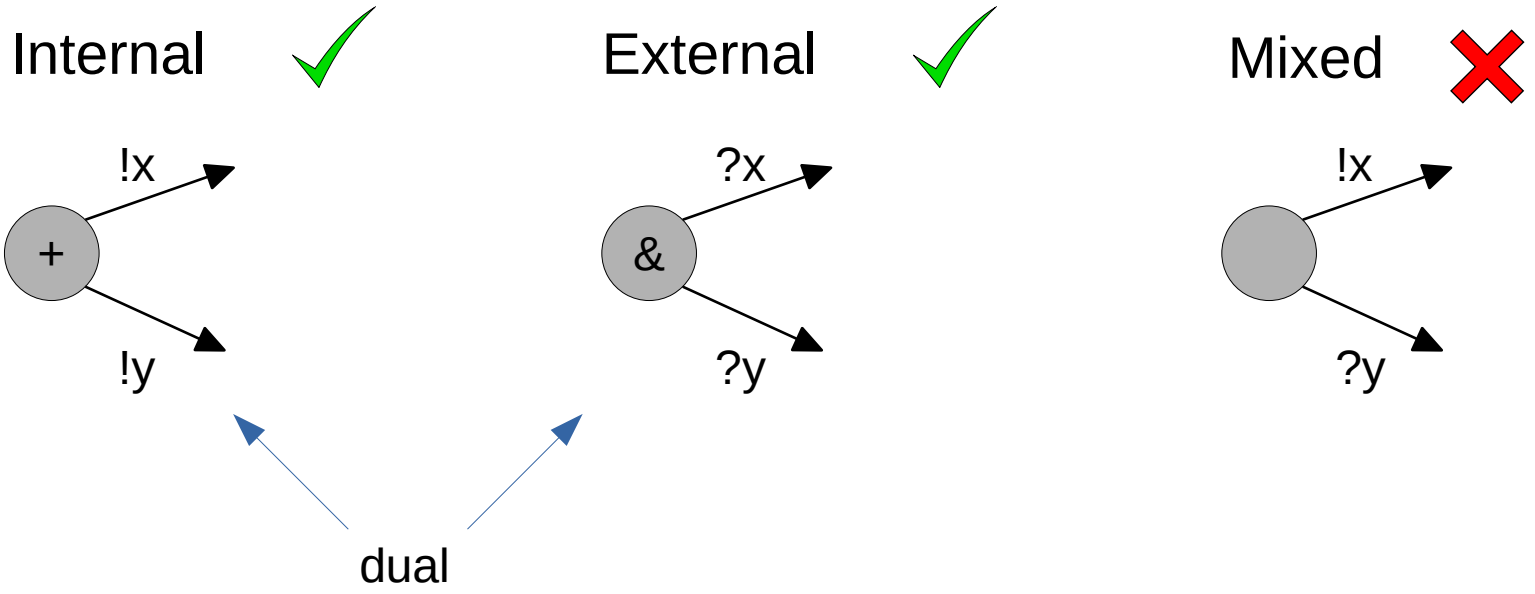


Binary session types (BST)

Idea: one side of the channel determines the operation on the other side.
1) get the operations for A
2) compute the operations for B using duality
3) check B executes the specified operations

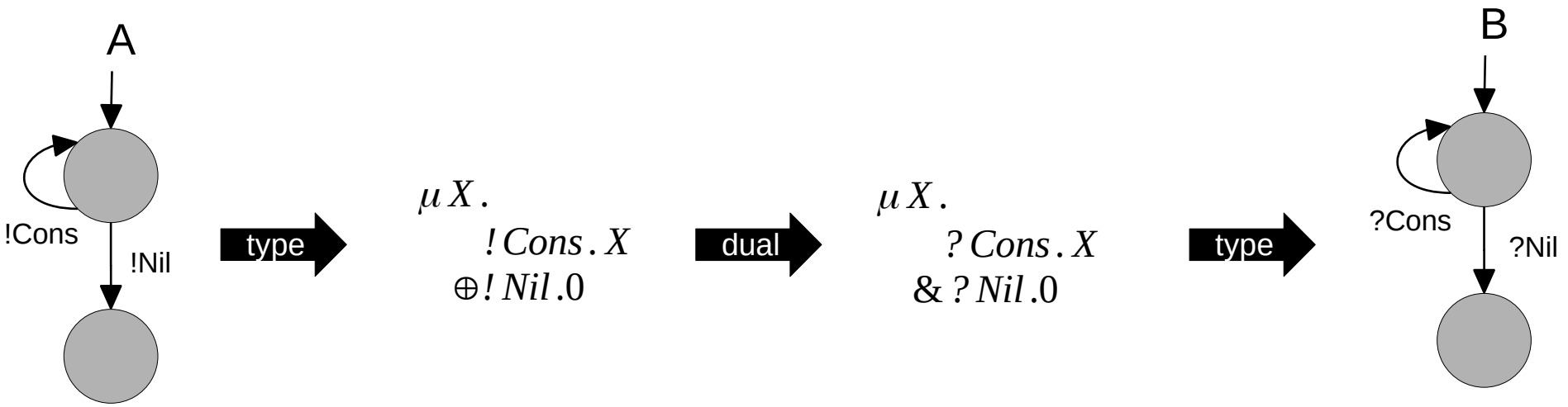


BST and choice



deterministic choice: all outgoing transitions have different labels

BST example with choice



BST and model checking

The syntactic restrictions on the protocols allowed by BST fall within the class of **half-duplex** protocols.

Half-duplex: buffer empty when communication switch direction

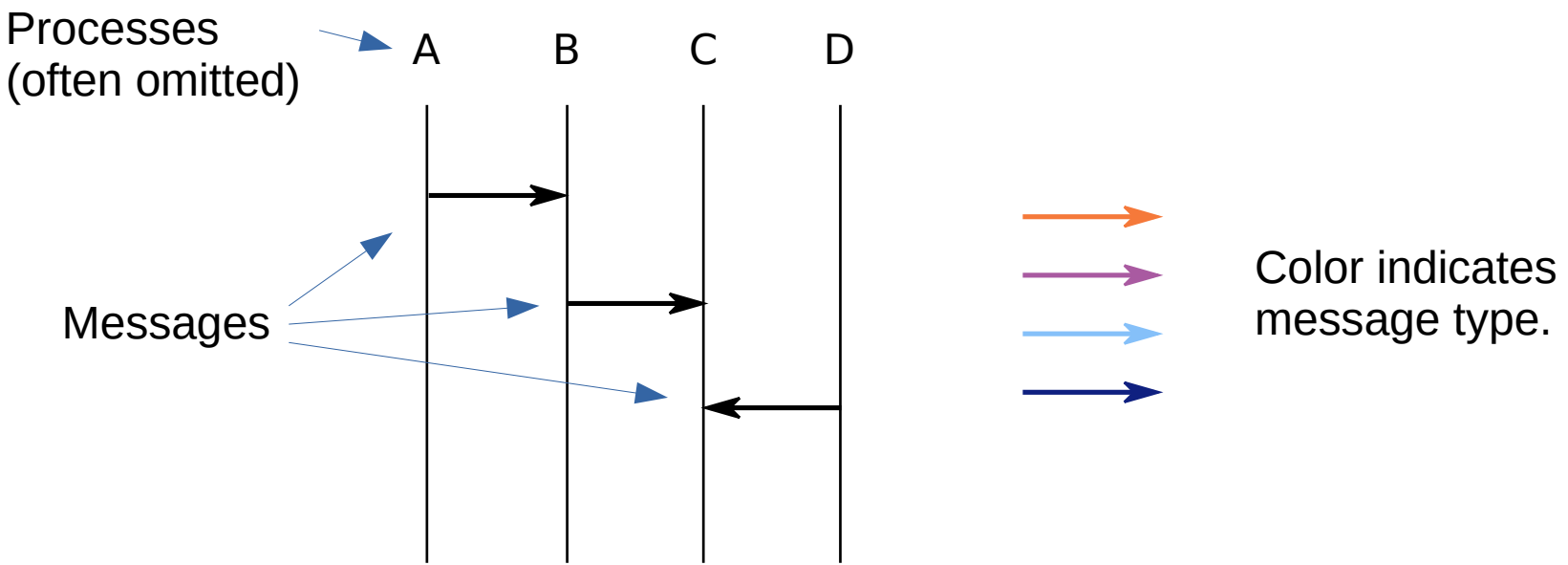
Half-duplex CSM with 2 processes are not Turing complete.

BST enforce this restriction on the protocol and runs the same on half- and full-duplex systems.

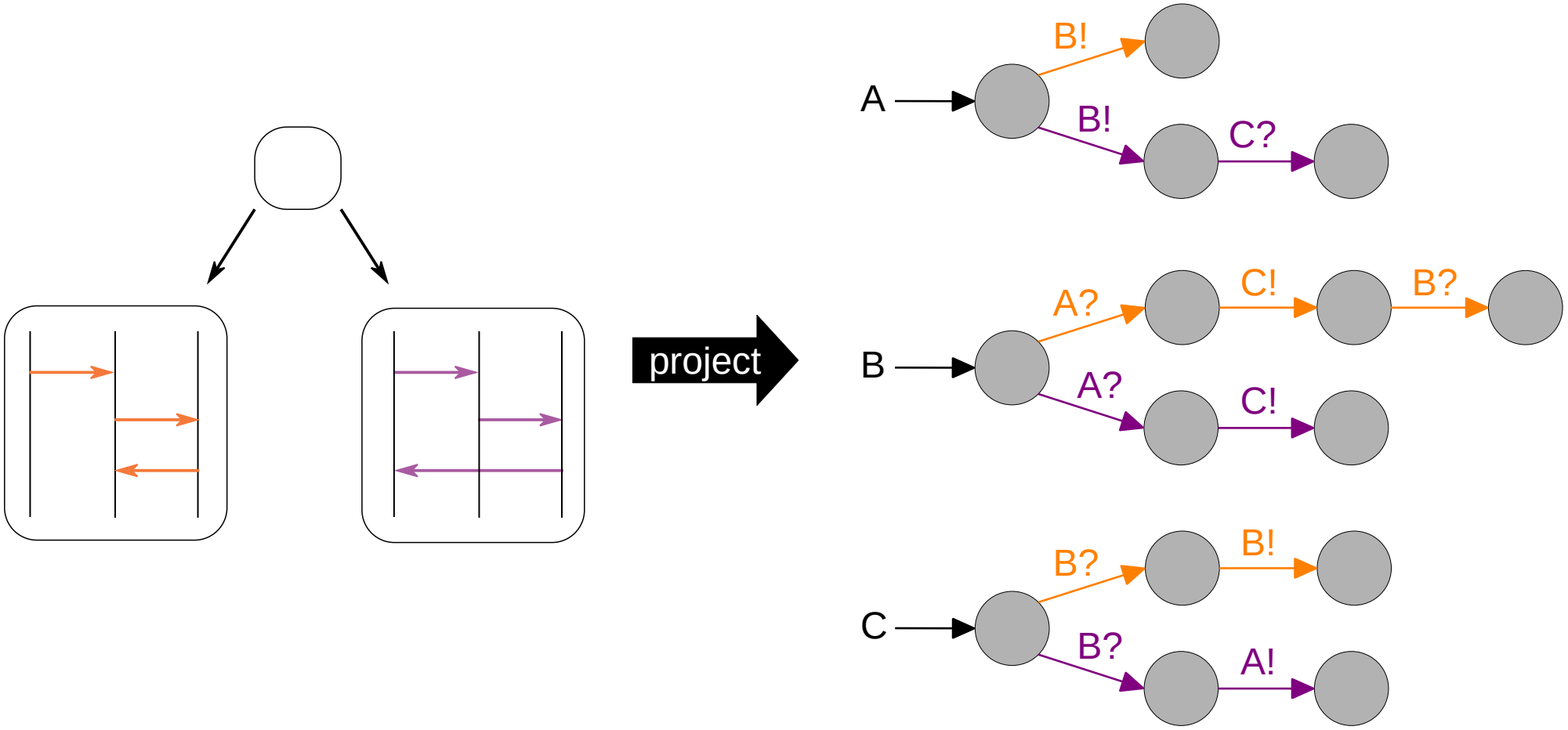
More than two processes...

- 2 processes: well understood
- 3+ processes: badly understood
  - Projection of global description is lossy. It can introduce non-determinism.
  - Messages sent by different processes are independent.

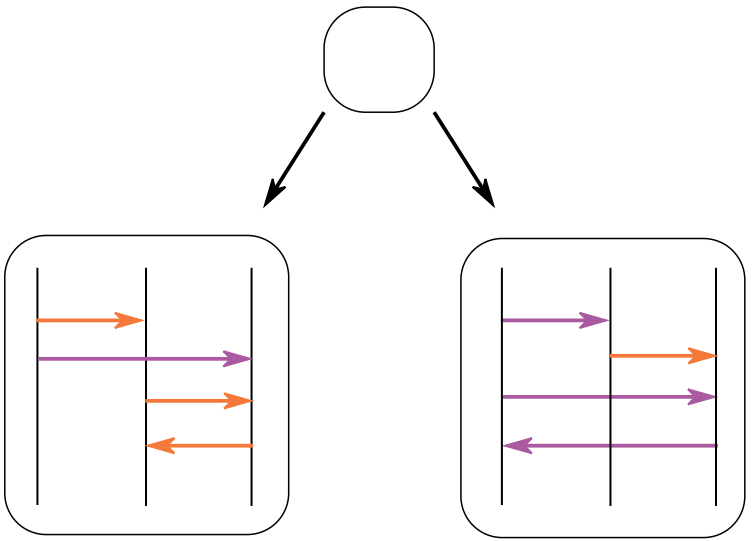
High level message sequence charts notation



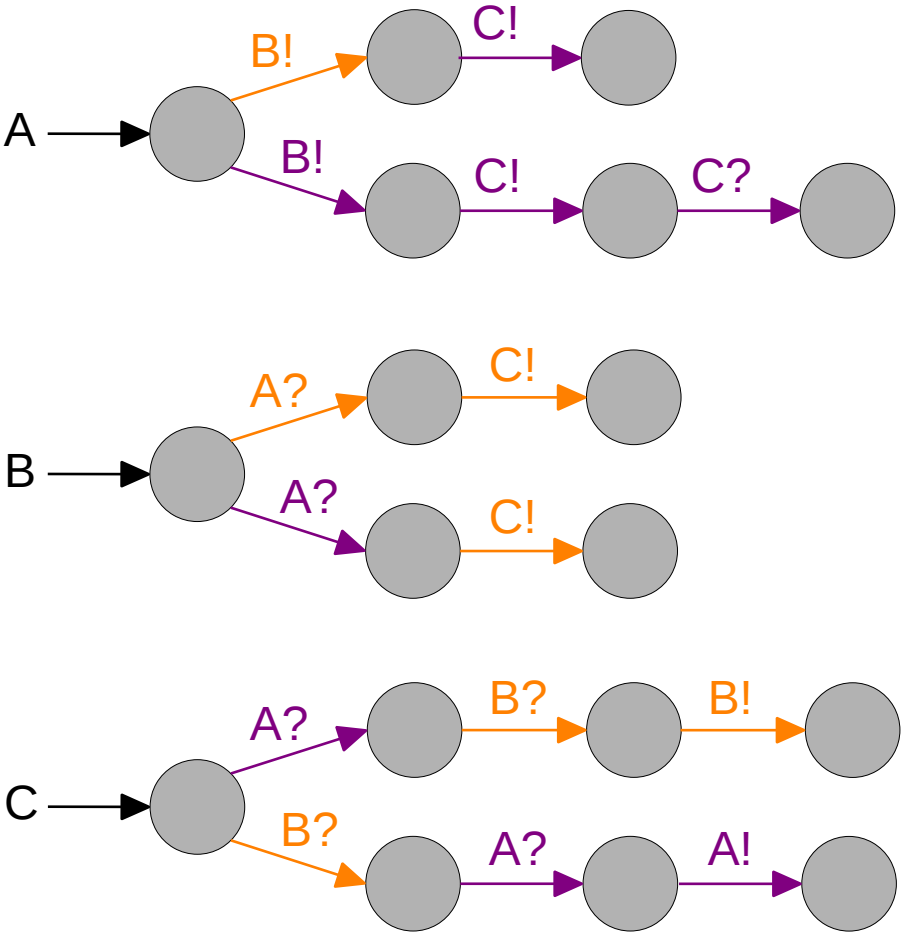
From duality to projection



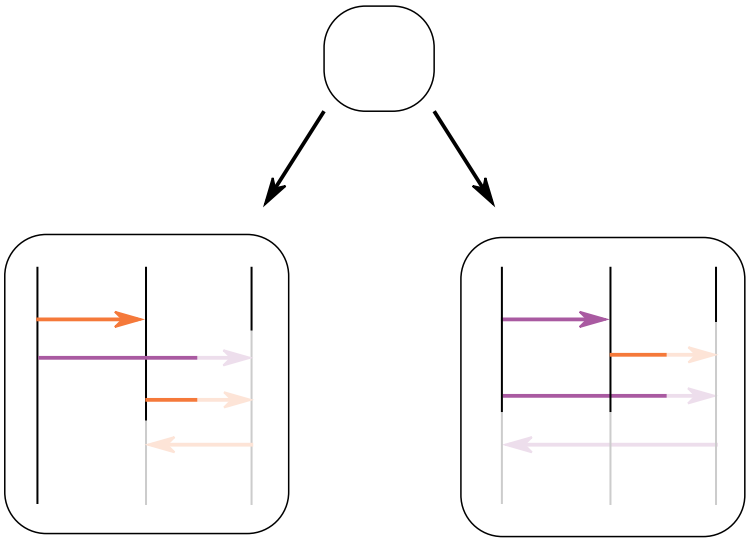
Another example



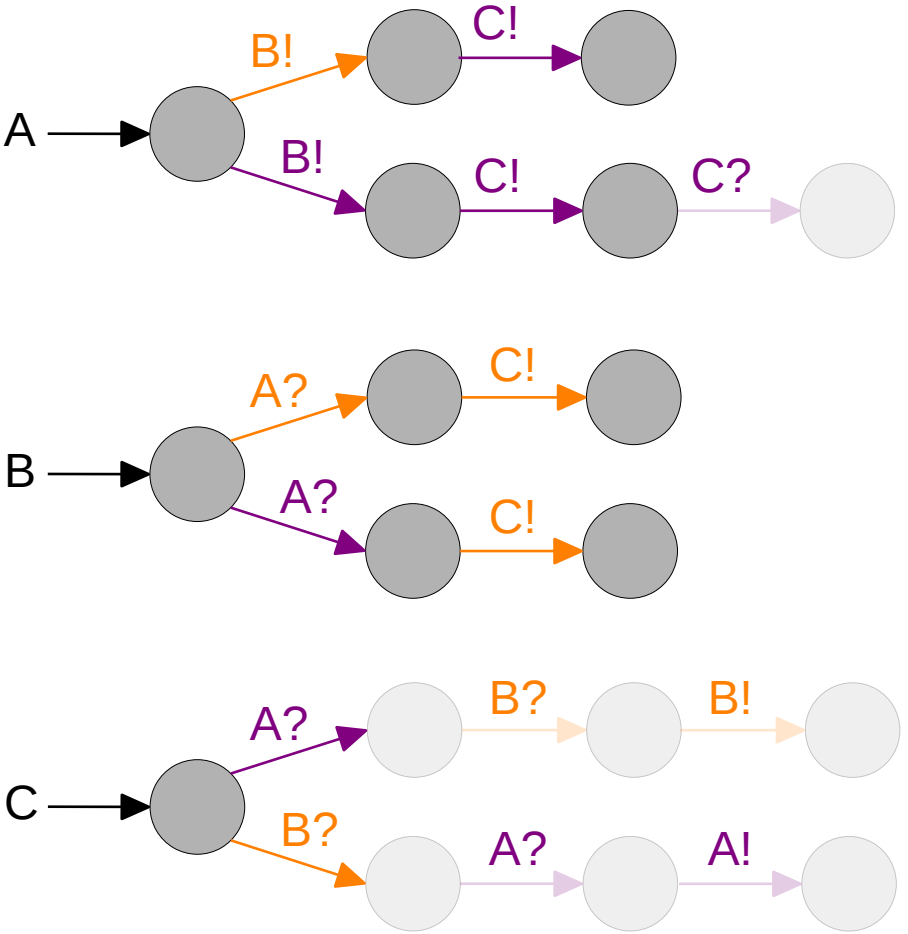
project



Which is wrong



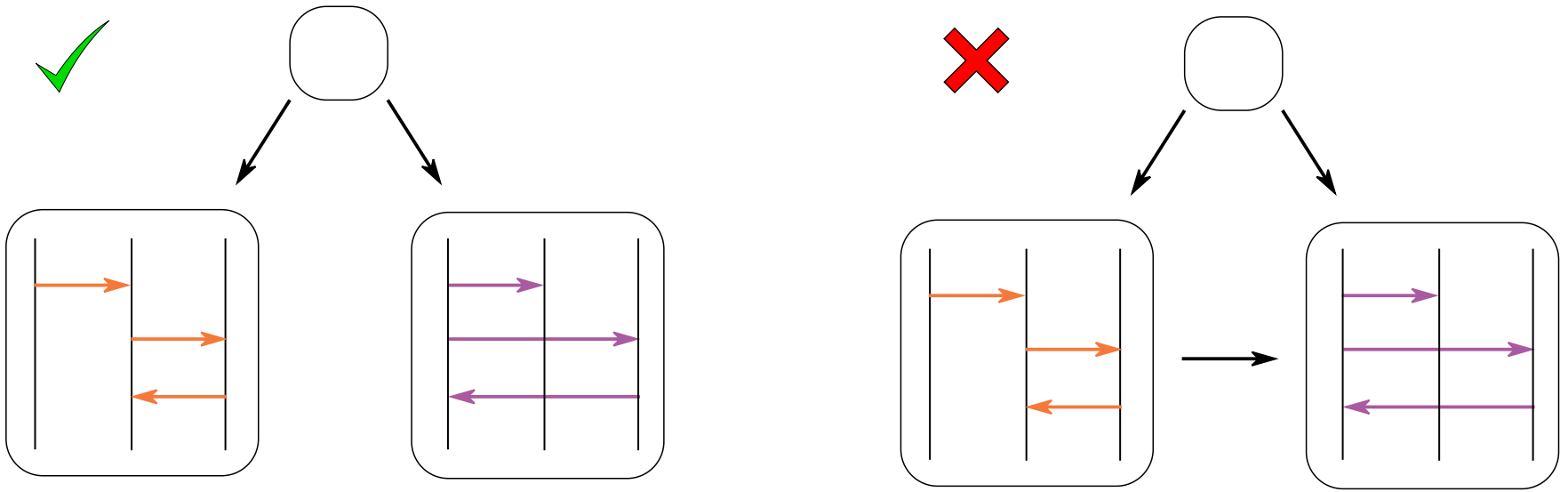
project



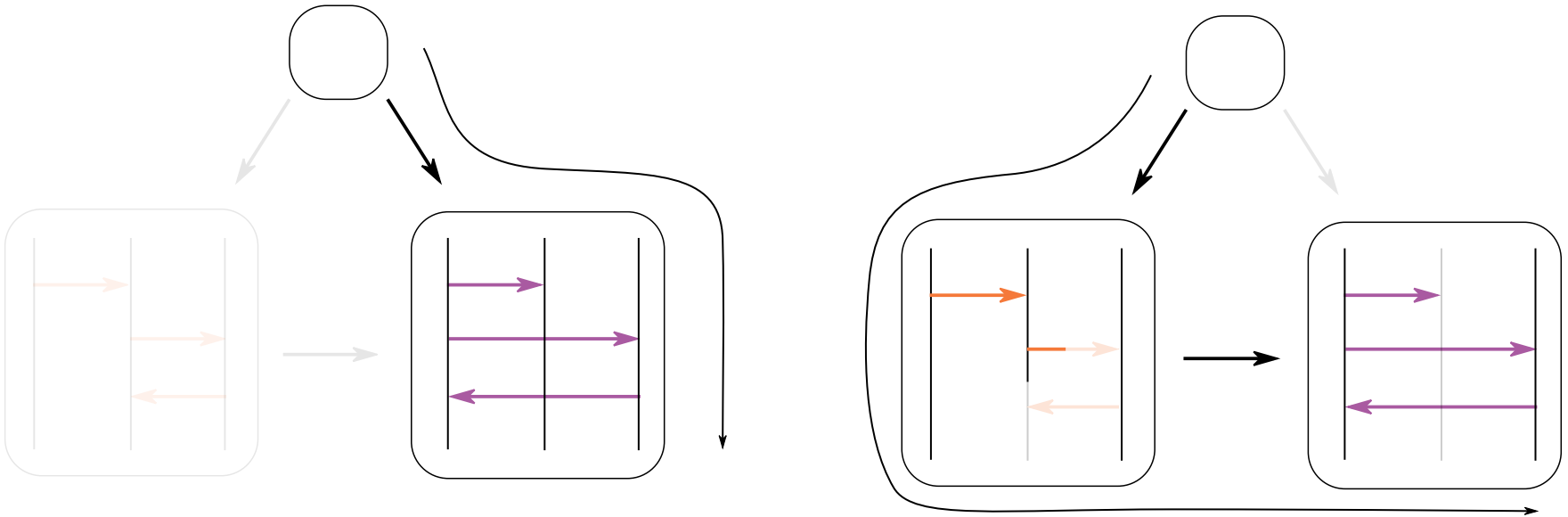
The messages from A,B are independent.
C cannot rely on their ordering.

No compositionality

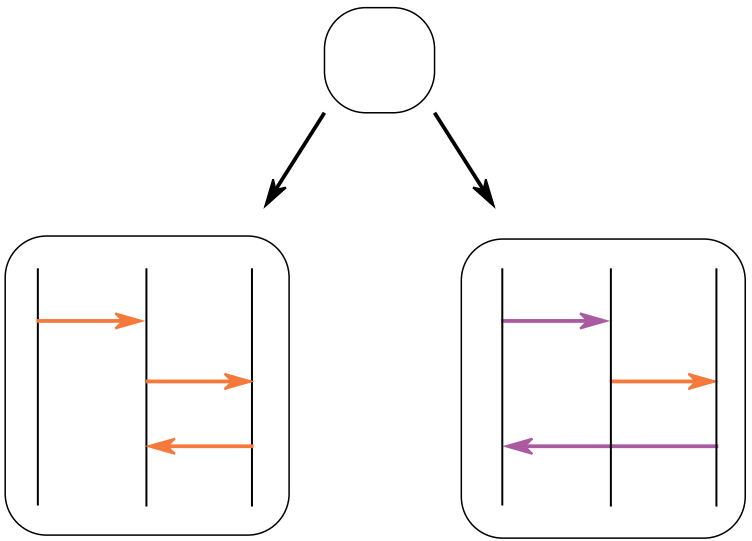
Composing two correct subprotocols can result in an incorrect one...



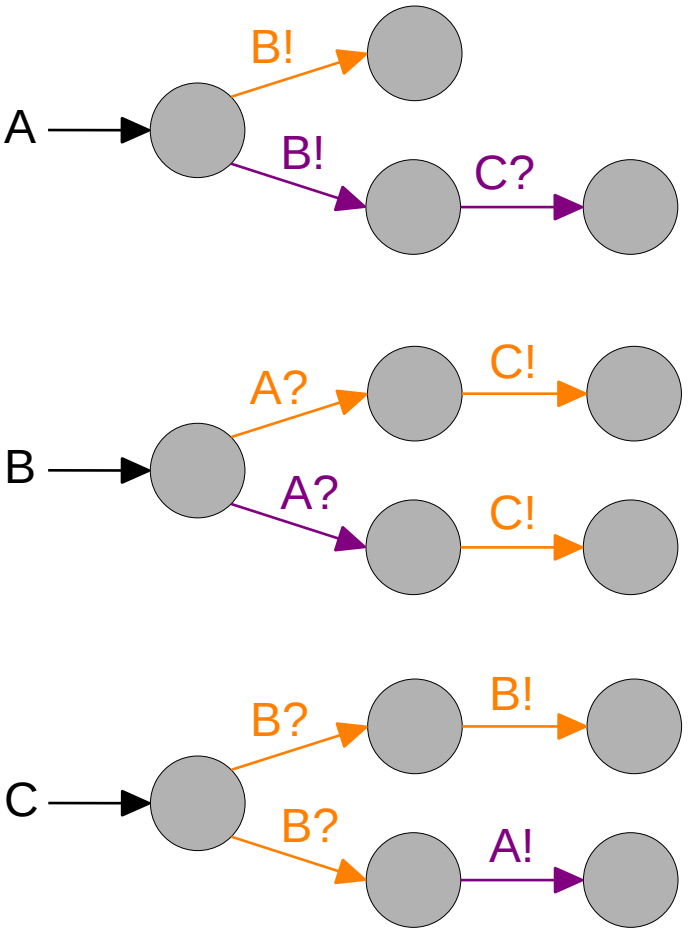
Counterexample



Determinism does not carry over



project



A makes the choice,
B is deterministic,
but C is not...

HMSC vs MST

HMSC

- Compact notation for CSM
- Verification based on state space exploration

Multiparty session types (MST)

- Try to generalize the ideas of BST
- Verification based on proof search

HMSC vs MST*

HSMC

- Permissive choice
- Mostly undecidable

  (unfold loops to get all behaviors)

Multiparty session types (MST)

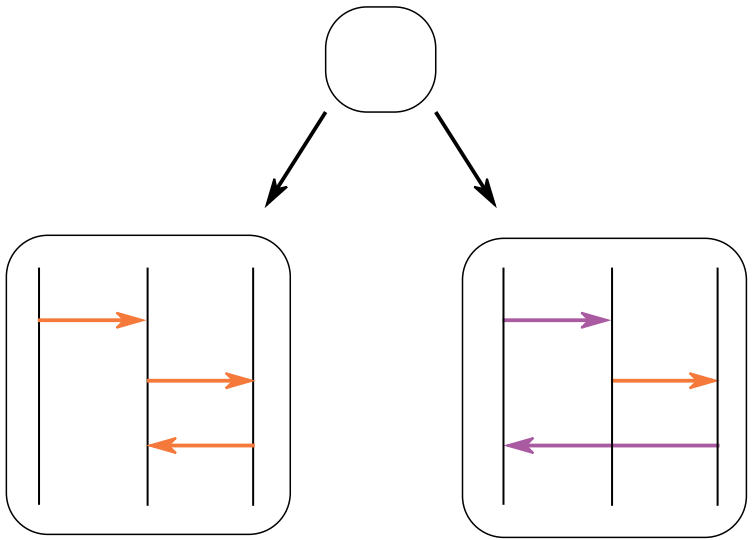
- Restricted choice
- Works when it works

  (proof search without unfolding)

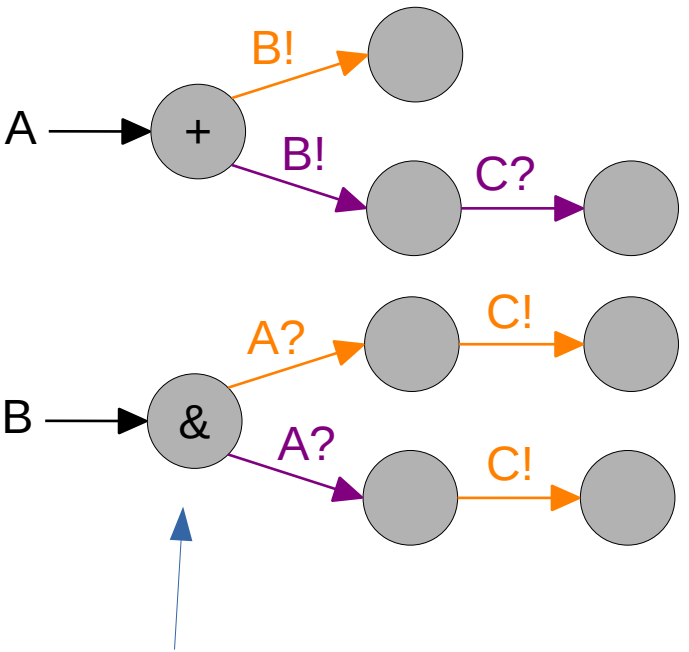
* comparison so simplified it is kind of wrong

MST in a nutshell (1)

Projects the protocol and checks the projection does not introduce unwanted nondeterminism.



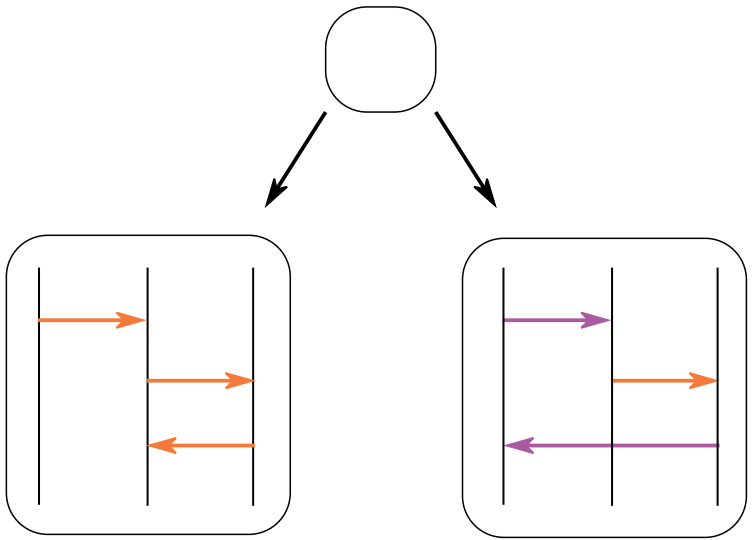
project



Keep track of internal/external choice

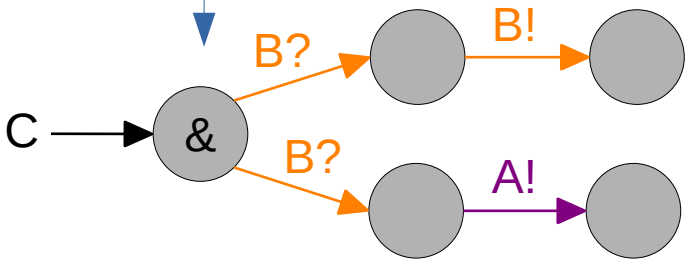
MST in a nutshell (2)

Projects the protocol and checks the projection does not introduce unwanted nondeterminism.



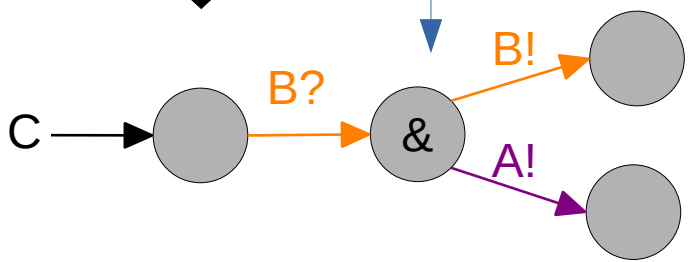
project

Same message on both side
Postpone choice resolution

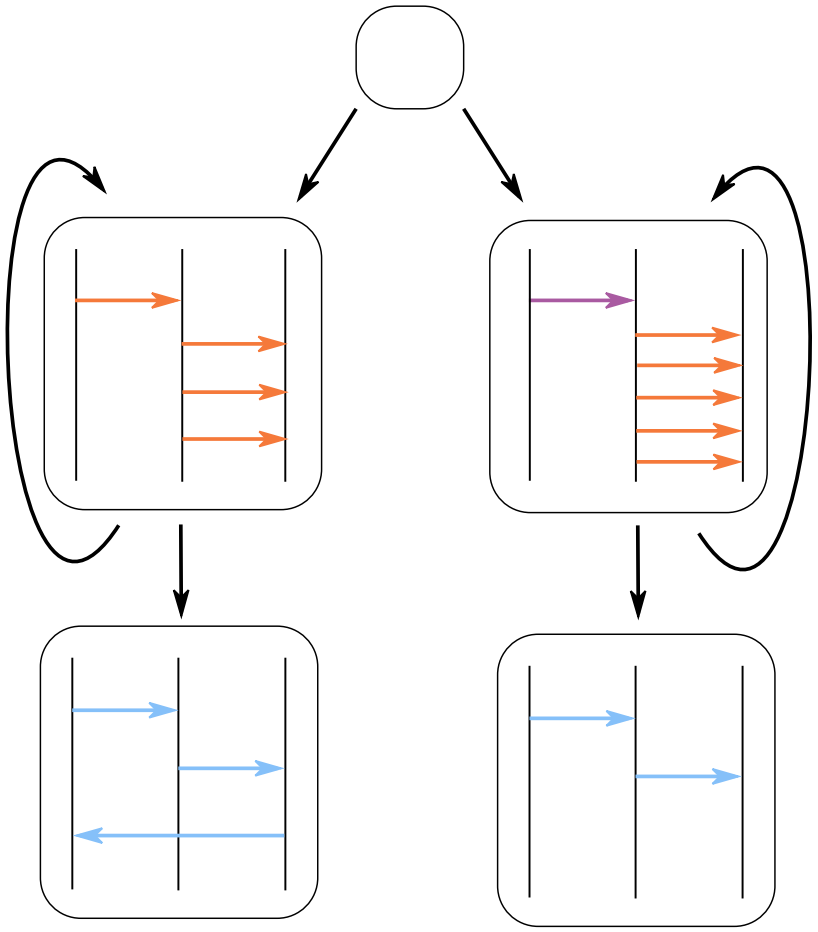


merge

External choice + send = error



Unfolding loops (1)

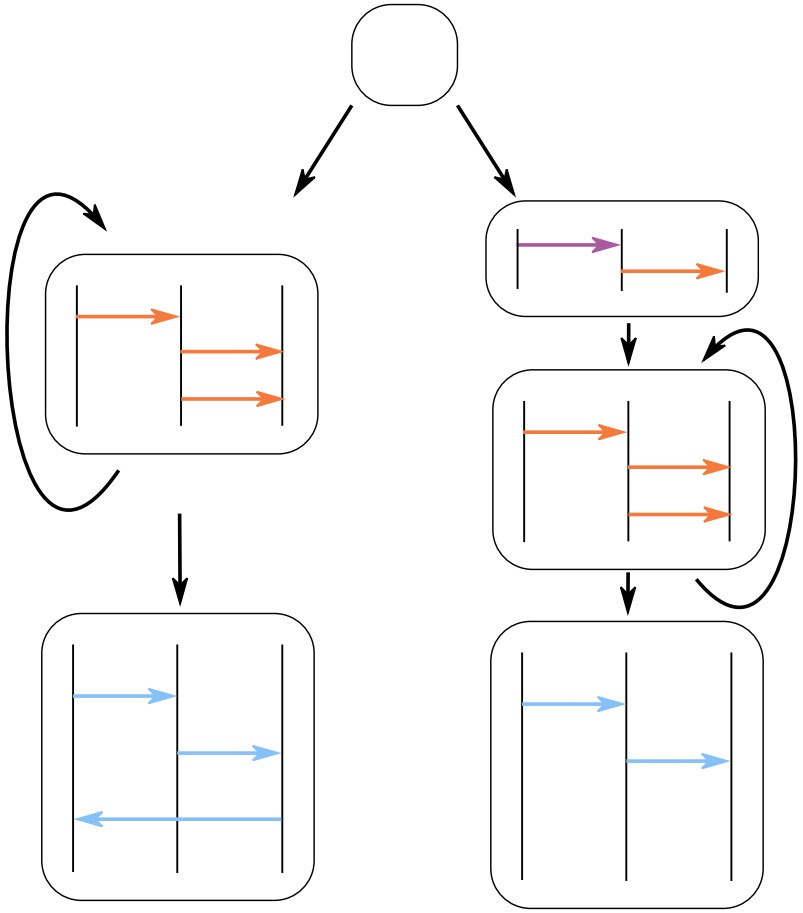


Loops can create an unbounded state space.

Model checking (BFS) finds an error unfolding the left 5x and the right 3x.

Rejected by MST without any unfolding, i.e., C does not learn the branch on the 1st iteration.

Unfolding loops (2)

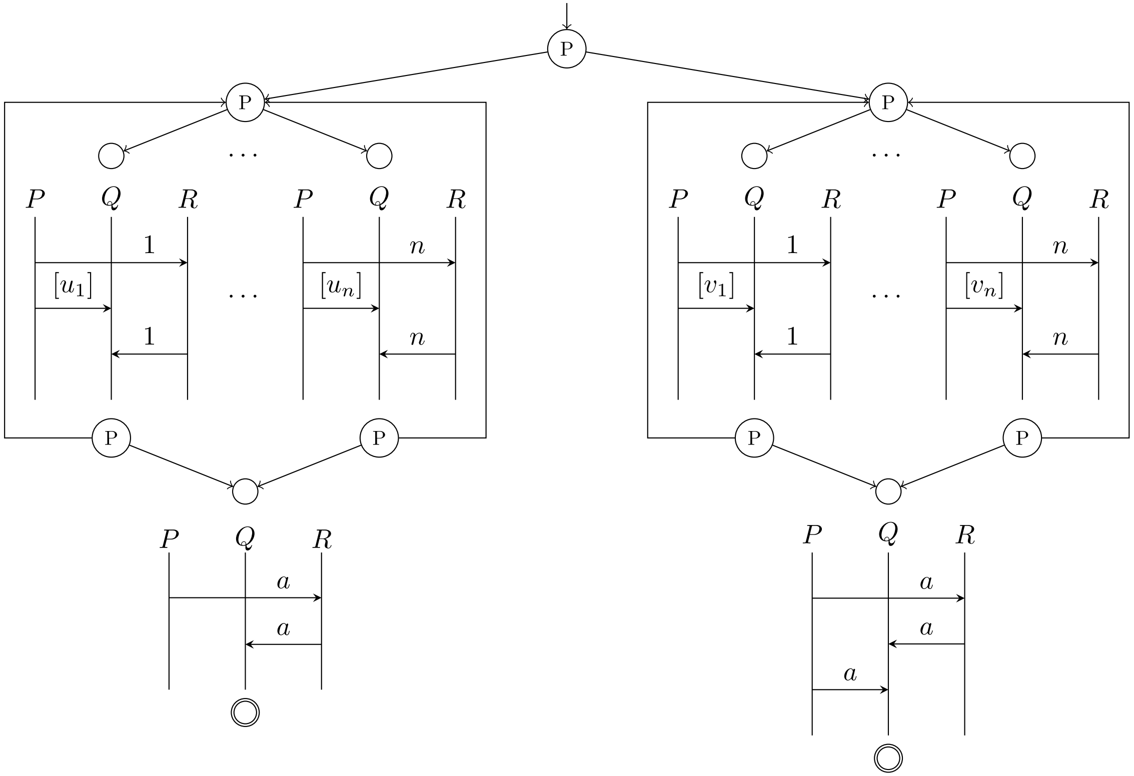


Here is a protocol that it implementable.

Model checking diverges.

MST rejects it.

Resolving choice with unfolding...



PCP encoding

... is hard.

Conclusion

- Good understanding of binary communication
  - Clean theory and (mostly) effective results
- More research needed in the multiparty case
  - Expressive and predictable
  - No need to be complete but interesting enough
- We only scratched the surface
  - safety properties only, no liveness yet
  - No mechanism to handle failure
  - Fixed set of processes
  - ...