On Choice in Protocol Specification

Generalizing Projection in Asynchronous Multiparty Session Types
CONCUR 21, with Rupak Majumdar, Madhavan Mukund, and Felix Stutz

On Channel Use in Protocols and General Point-to-Point Communication
under submission, with Felix Stutz

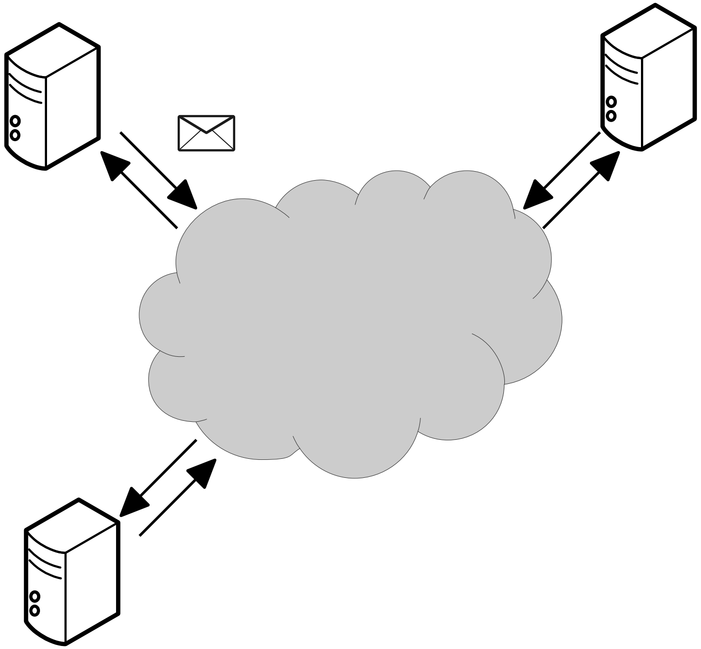
Damien Zufferey
MPI-SWS
2021.09.30

Overview

- Detour: why it is hard to reason about CSM?
- Binary session types and half-duplex channels
- Binary to multiparty session types
- Directed to generalized choice
- Future work

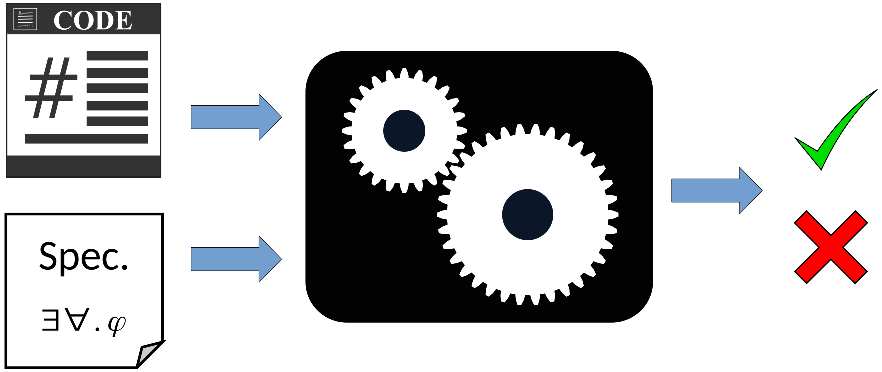
What is this about ?

Distributed systems
(message-passing)

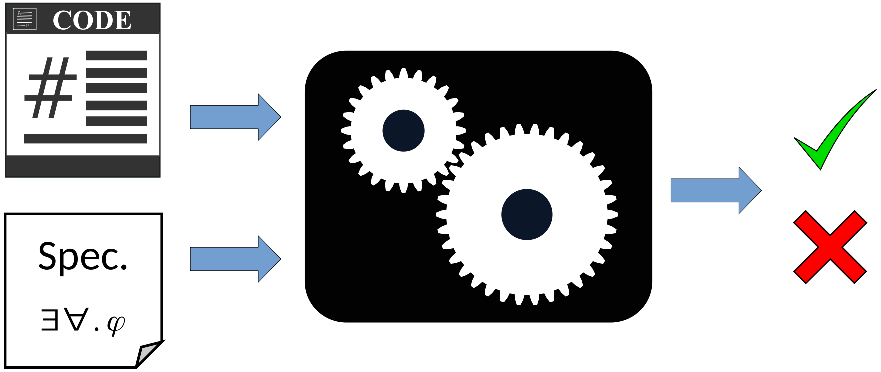


+

Verification



What to Verify?



A communication protocol is correctly implemented:

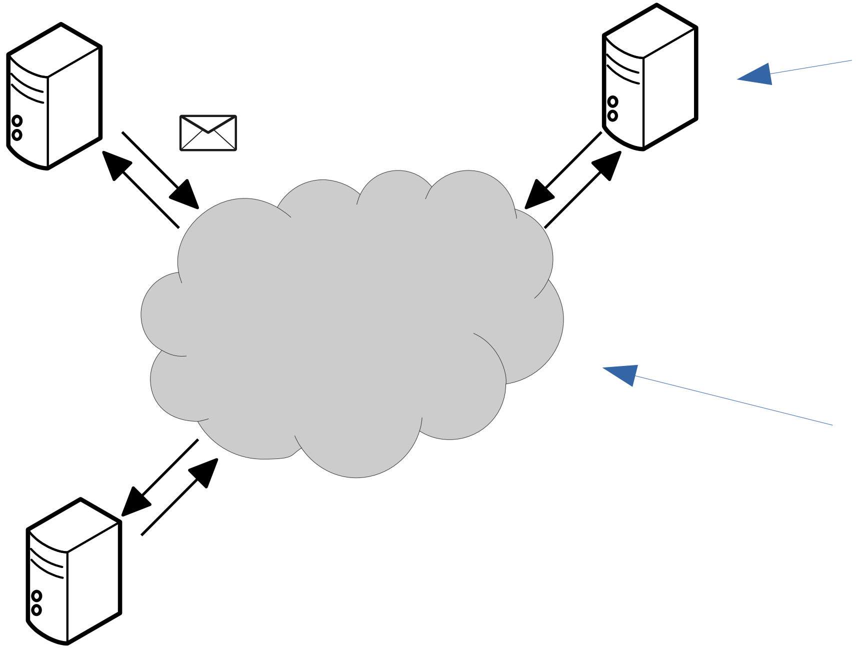
- messages follow a prescribed order
- deadlock-free
- no dangling messages

Two Approaches

Type System	**Model Checking**
based on proof theory	based on automata theory
+ effective	- expensive
- not ‘typeable’ ≠ incorrect	+ counterexample
research toward more expressive	research toward more specific

We will try to see both approaches for verifying communicating systems.

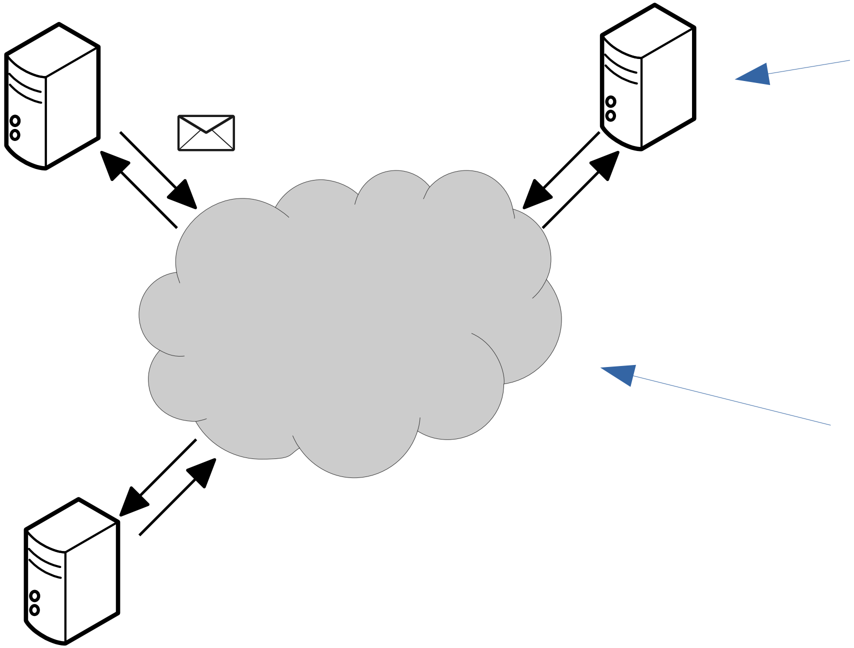
Model for the Software



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

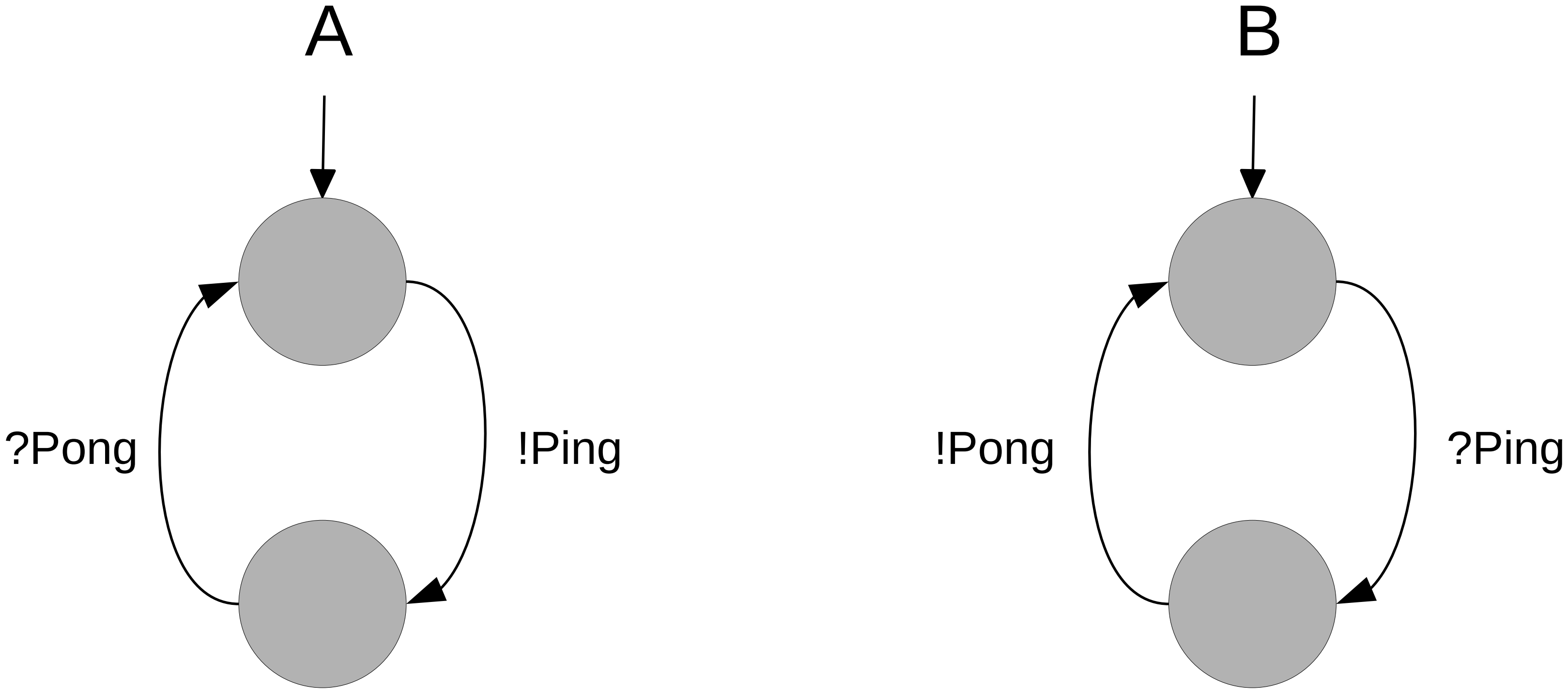
Communicating State Machines (CSM)



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

“Hello World” of Message-Passing



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

CSM: Expressive Power

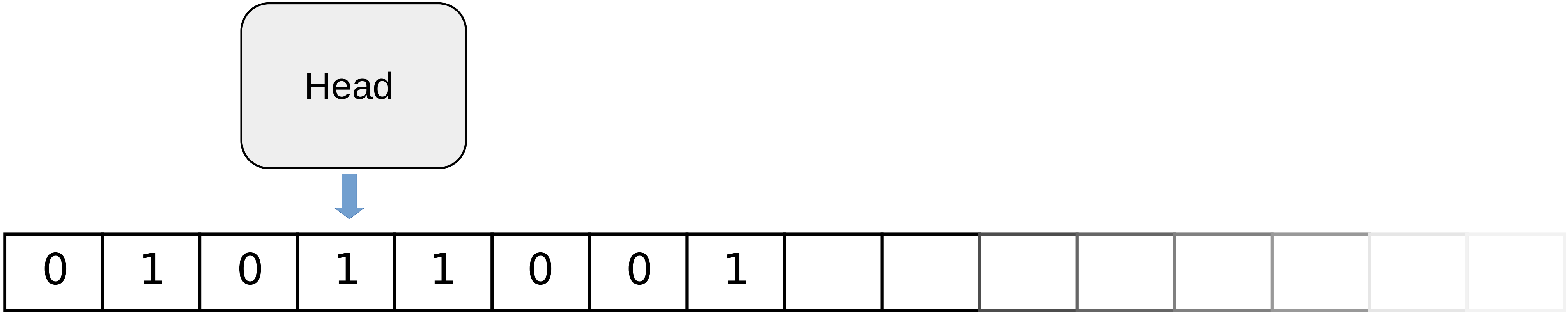
- Knowing the expressive power tells us what method can or cannot be applied.

  (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



Head

0	1	0	1	1	0	0	1								

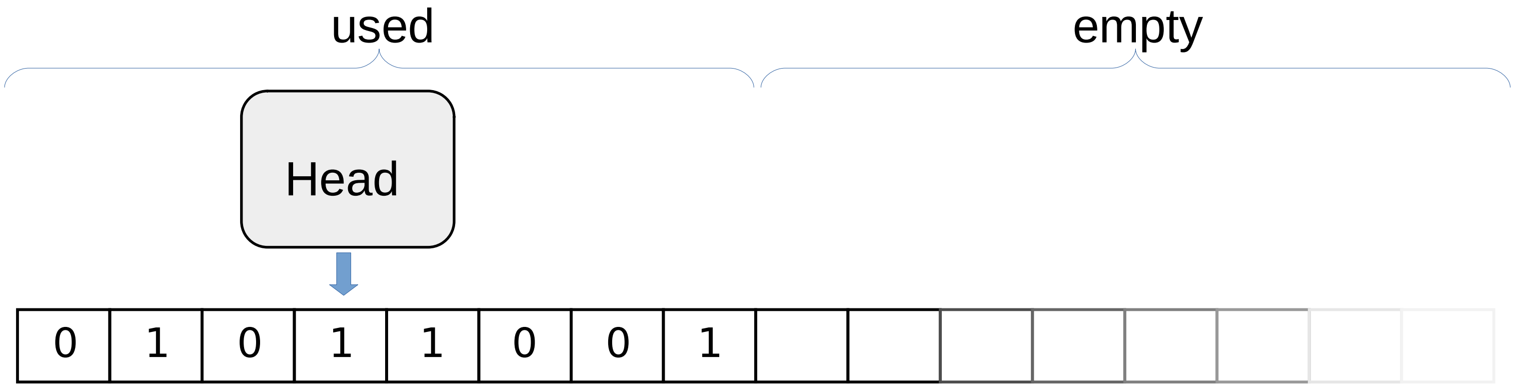
TM → CSM

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

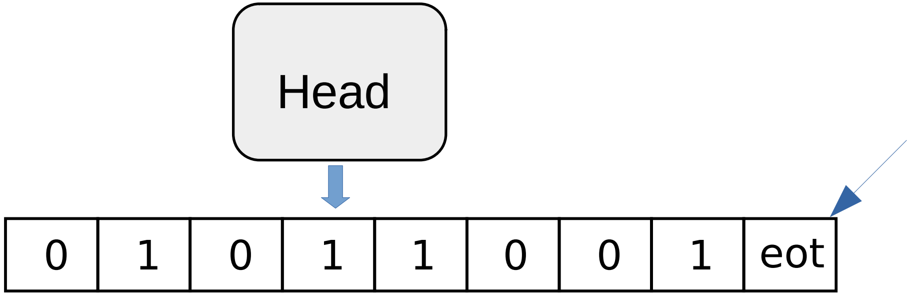


used

empty

Head

0	1	0	1	1	0	0	1								

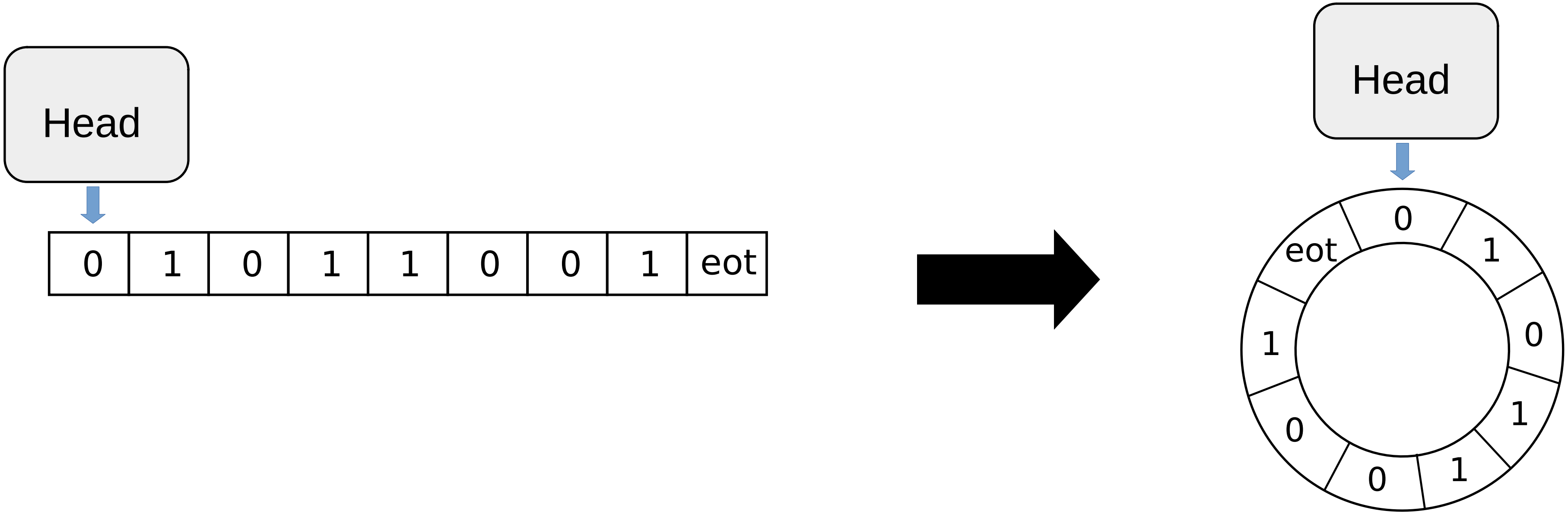


Head

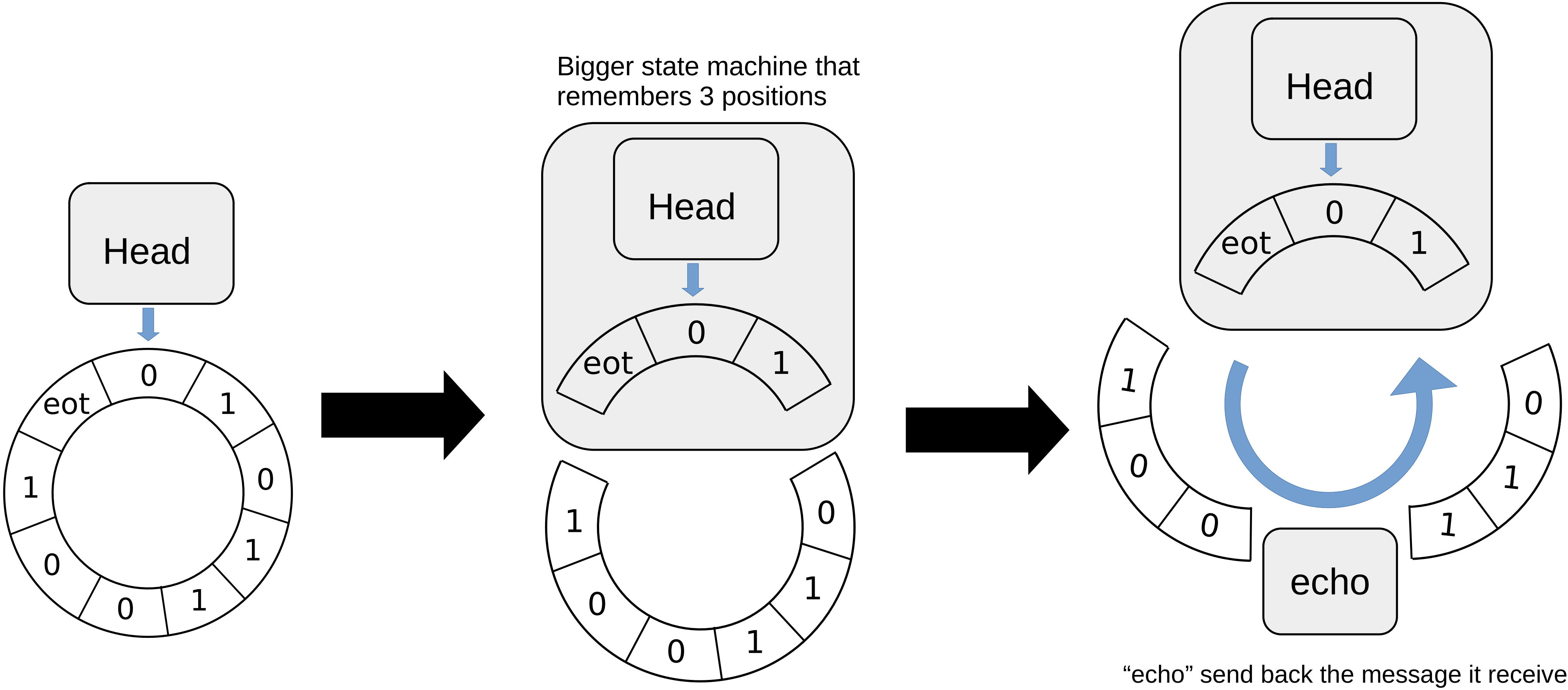
0	1	0	1	1	0	0	1	eot

“end of tape” symbol,
extend the tape on demand.

TM → CSM



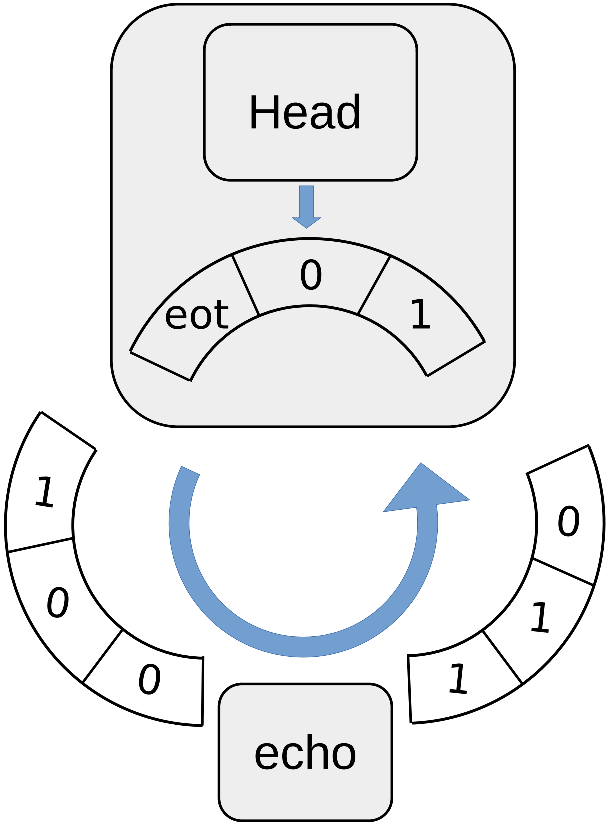
TM → CSM



Bigger state machine that remembers 3 positions

“echo” send back the message it receives

TM → CSM



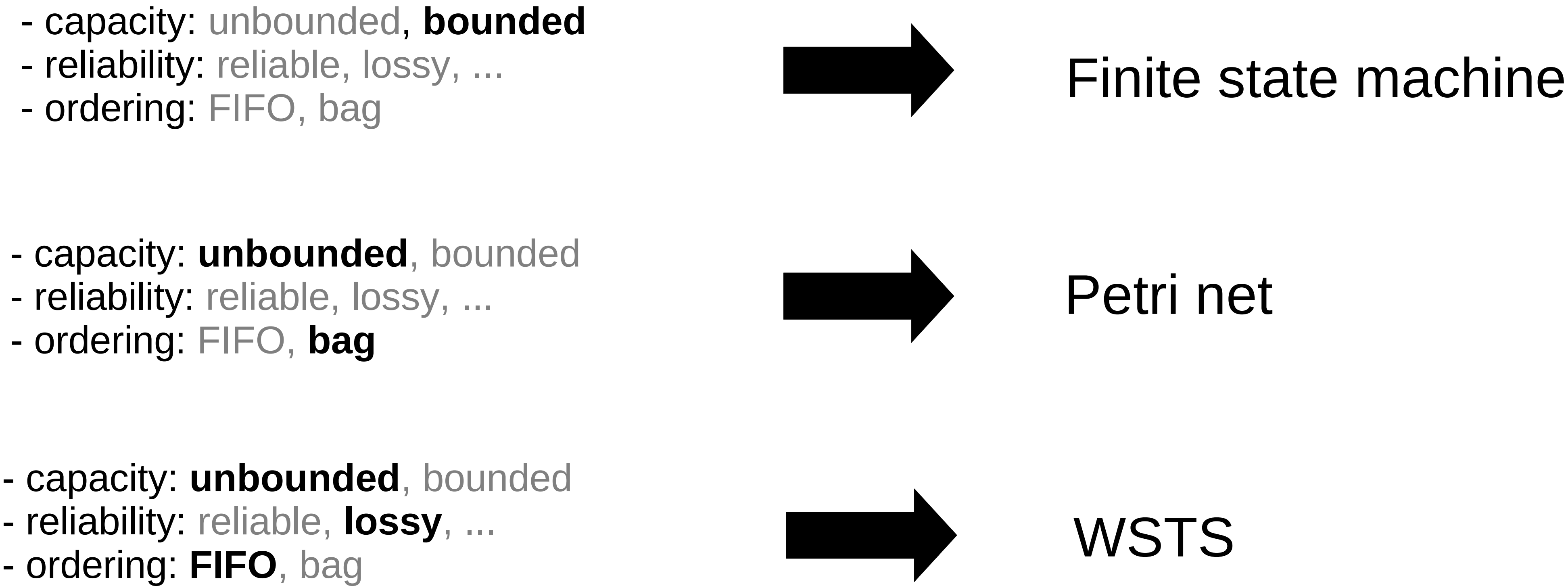
Emulating TM operations:

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

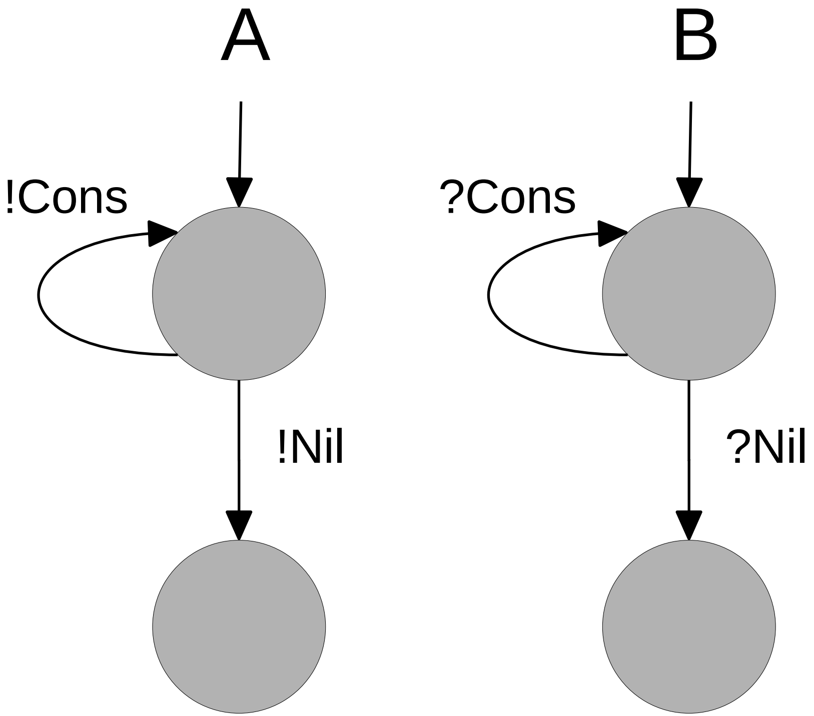
- capacity: **unbounded**, bounded
- reliability: reliable, lossy, ...
- ordering: FIFO, **bag**

Petri net

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

WSTS

Memory vs Communication Slack



Some protocols using unbounded channels can be harmless.

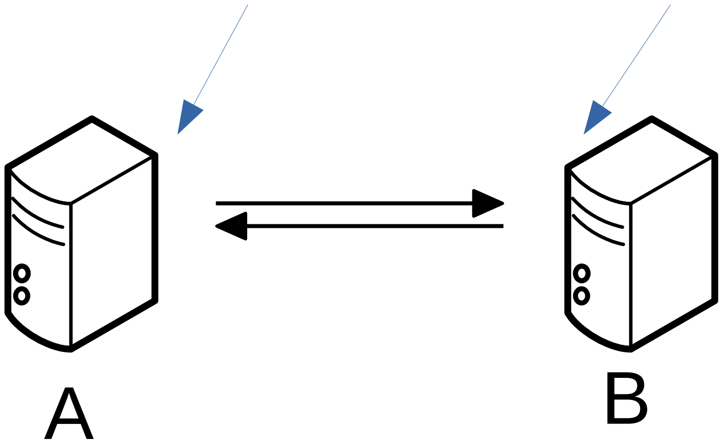
What is the difference between

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

Looking at the Channels

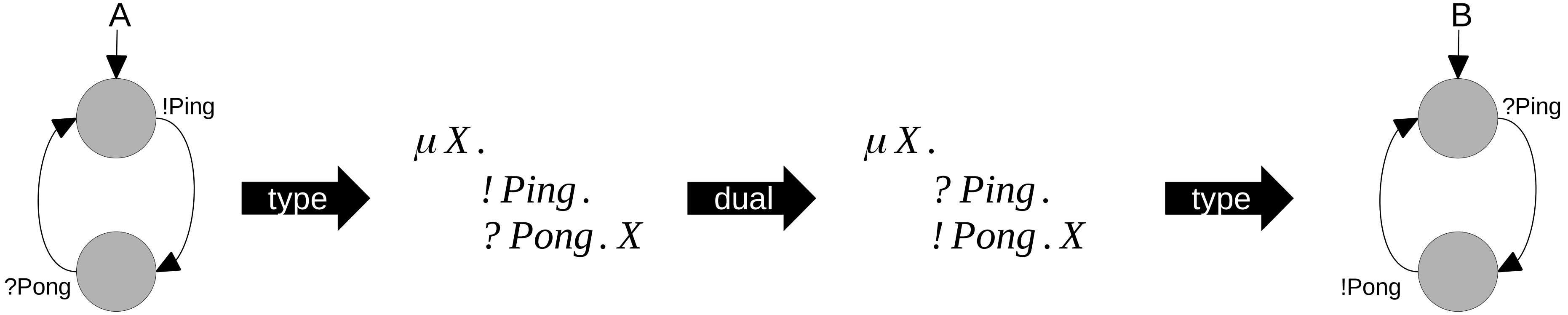
Operations on channel endpoints are **dual**.

send here = receive there



Binary Session Types (BST)

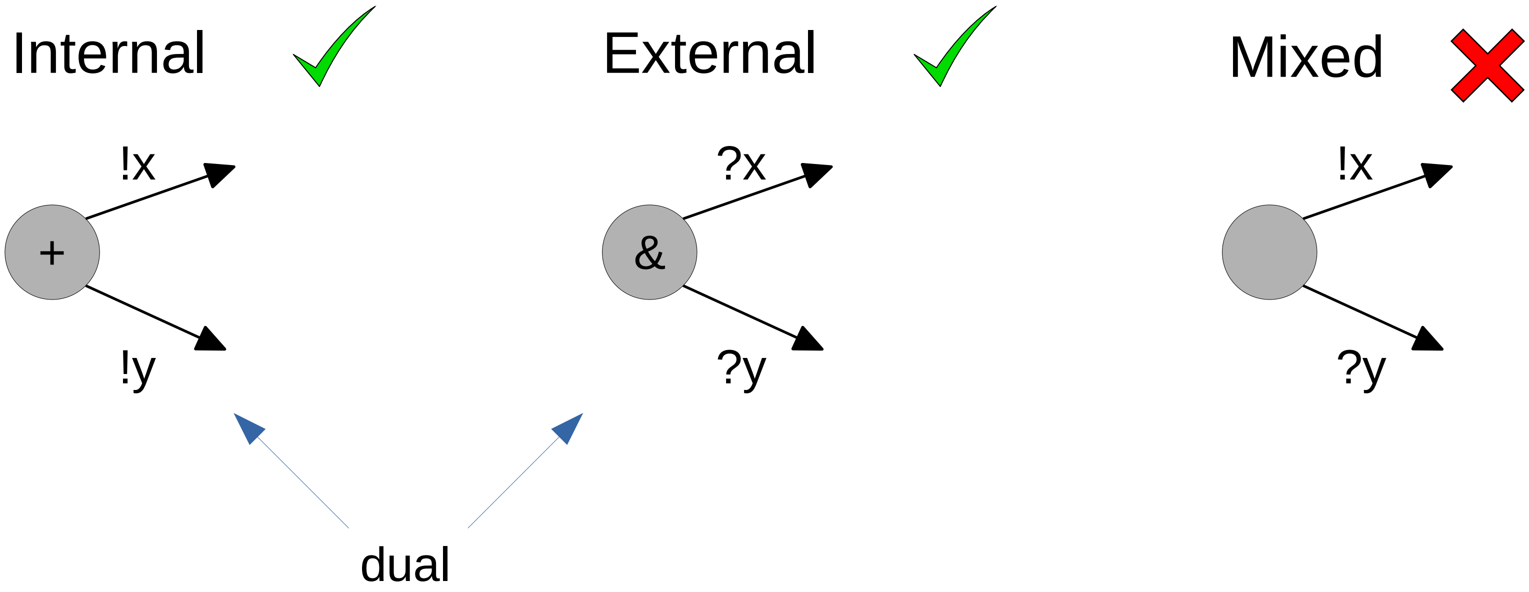
Idea: one side of the channel determines the operation on the other side.
1) get the operations for A
2) compute the operations for B using duality
3) check B executes the specified operations



$$\mu X.\ !Ping.\ ?Pong.X$$

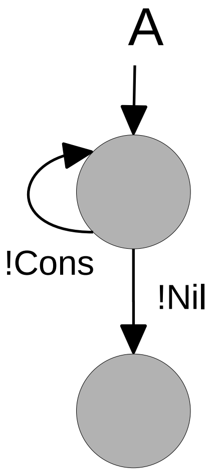
$$\mu X.\ ?Ping.\ !Pong.X$$

Duality in Choice



deterministic choice: all outgoing transitions have different labels

BST Example with Choice



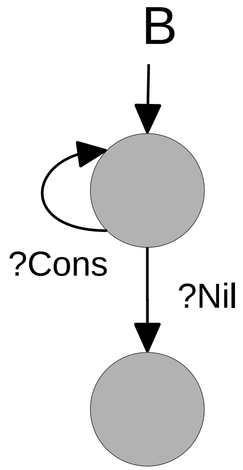
type

$$\begin{aligned} &\mu X. \\ &\quad !Cons.X \\ &\oplus !Nil.0 \end{aligned}$$

dual

$$\begin{aligned} &\mu X. \\ &\quad ?Cons.X \\ &\& ?Nil.0 \end{aligned}$$

type



BST and Model Checking

BST specifies only **half-duplex** protocols. [Cécé05]

Half-duplex:

- Buffers are empty when communication switch direction
- With 2 processes: not Turing complete

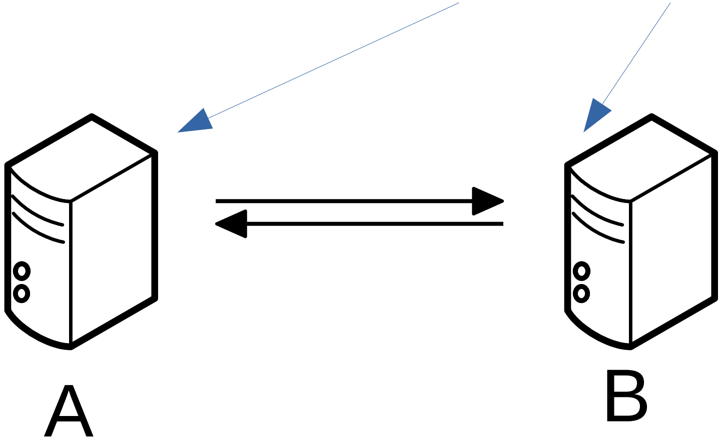
BST enforce this restriction on the protocol: runs the same on half- and full-duplex systems.

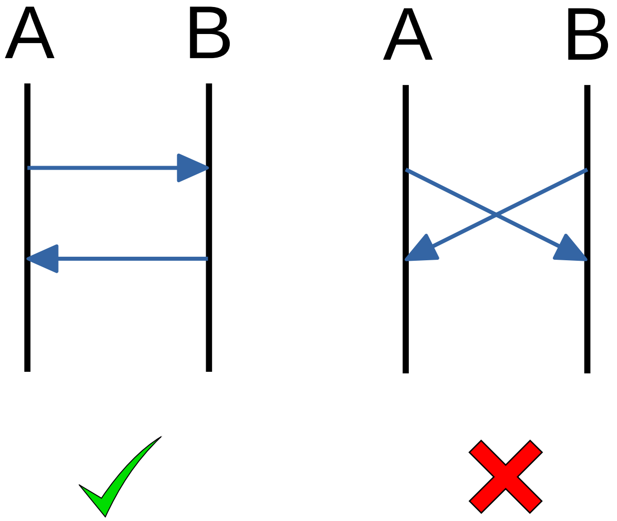
[Cécé05] Gérard Cécé and Alain Finkel. Verification of programs with half-duplex communication. Inf. Comput., 2005.

Duality and Message Spec.

Dual operations are easily **specified together**.

$A \rightarrow B{:}m \;\approx\; A!B(m)\;.\;B?A(m)$





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.

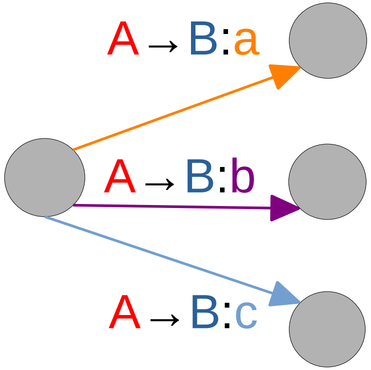
Binary to Multiparty

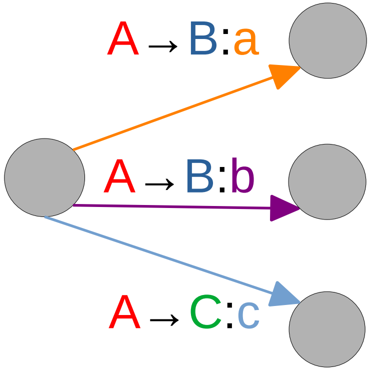
Find some restriction to keep protocol “well-behaved”.
Keep track of choices without getting confused.

- First with directed choice (classical MST)
- Then with generalized choice (CONCUR paper)

Directed vs Generalized Choice

Globally



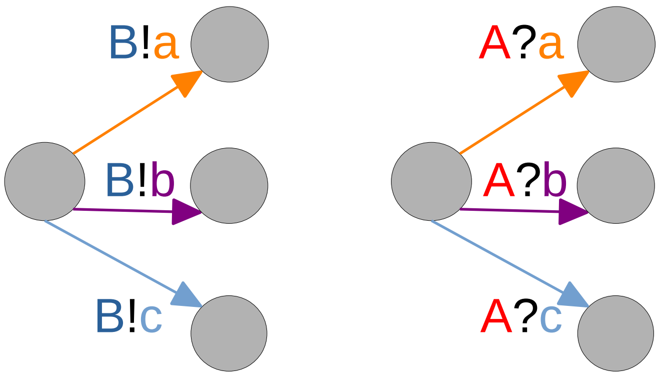


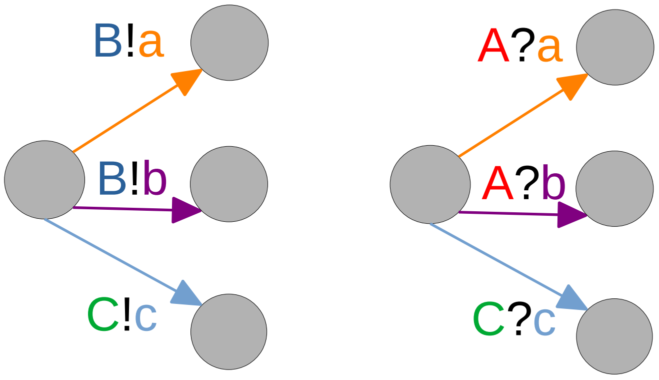
A,B are constant.
Only the messages (a,b,c) change

Sender A is constant.
The receivers (B,C) and messages changes

Directed vs Generalized Choice

Locally

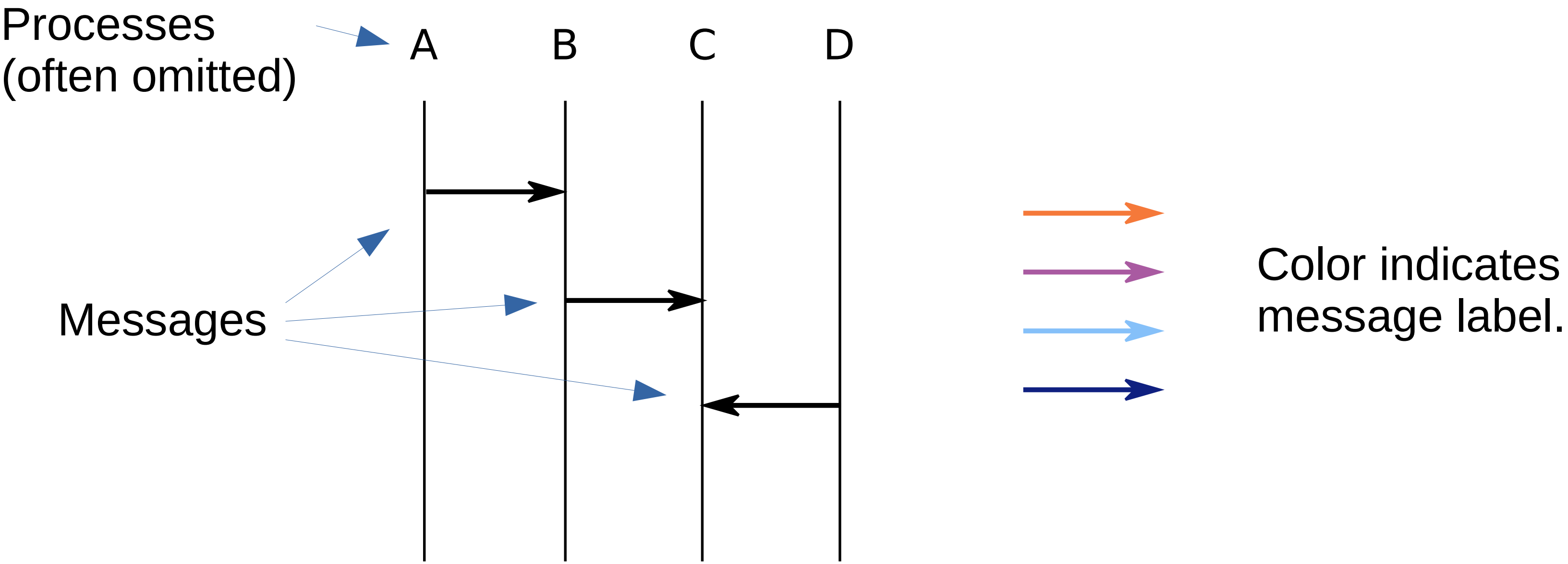




A,B are constant.
Only the messages (a,b,c) change

Everything can change.

High Level Message Sequence Charts Notation

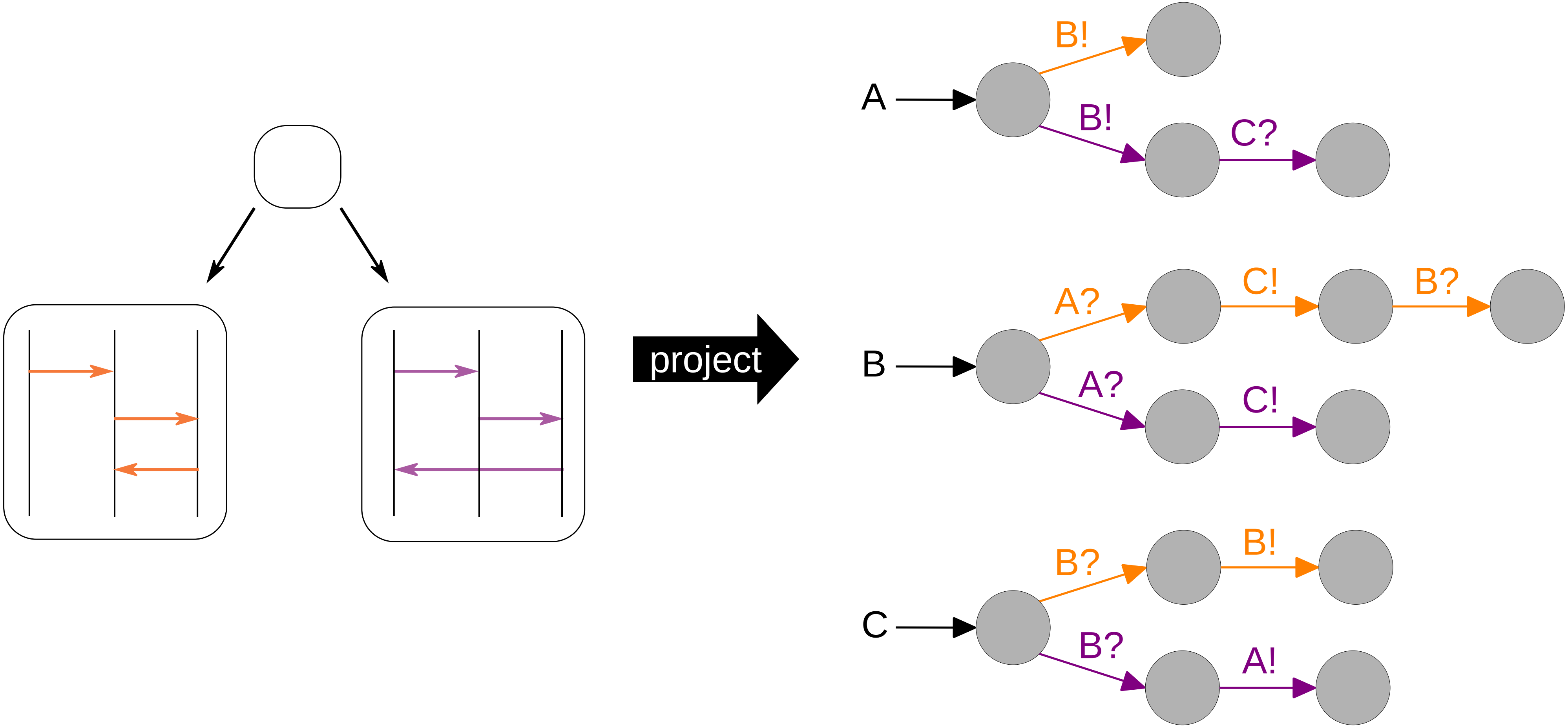


Processes
(often omitted)

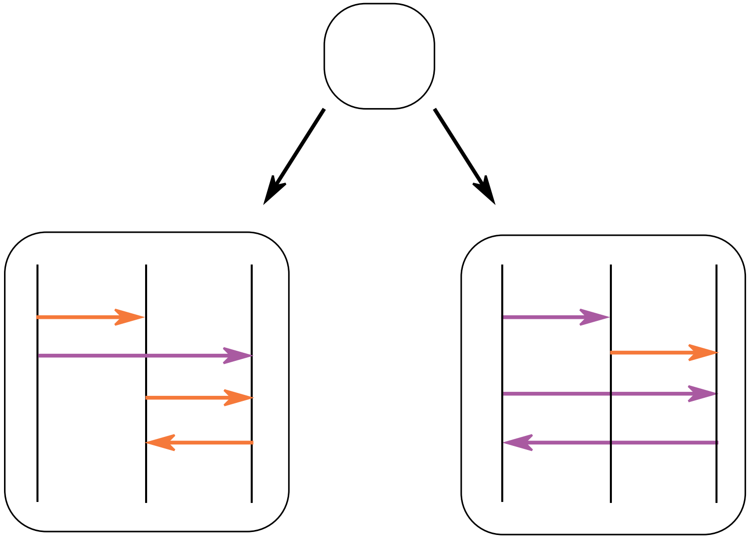
Messages

Color indicates
message label.

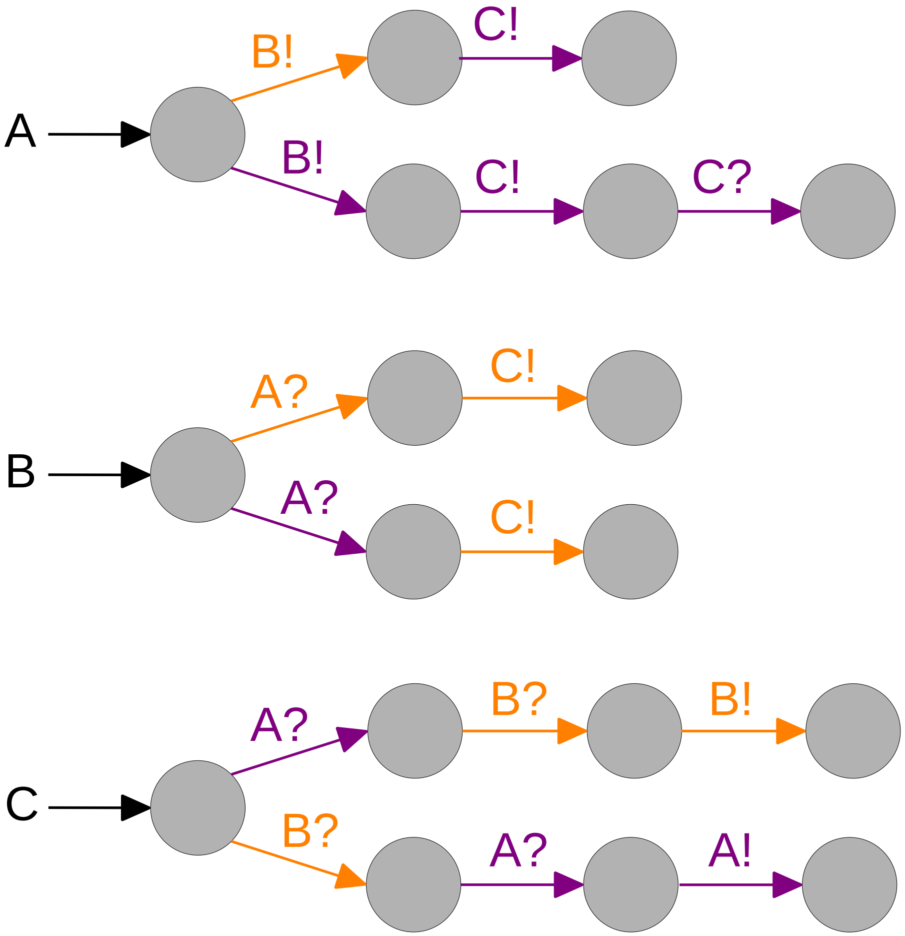
From Duality to Projection



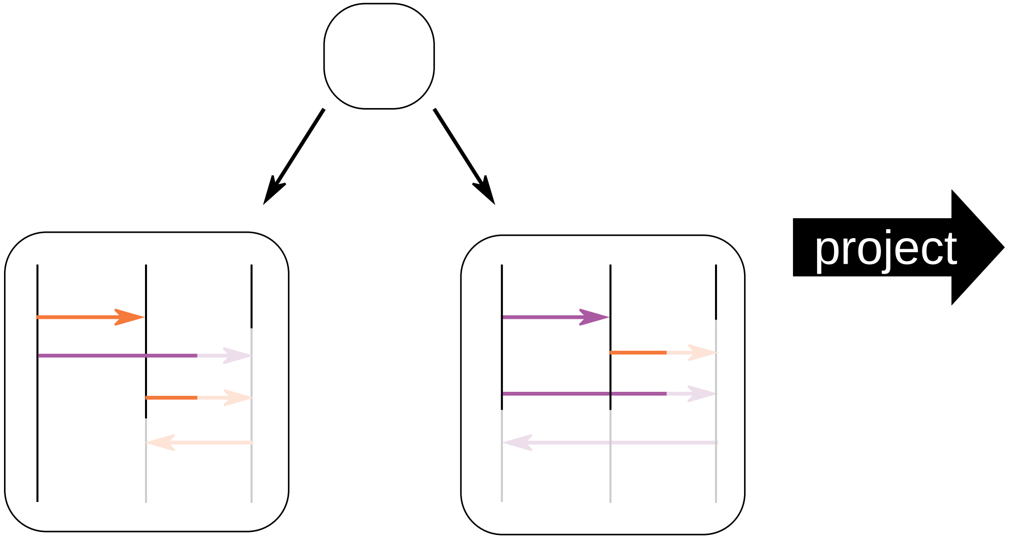
Another Example

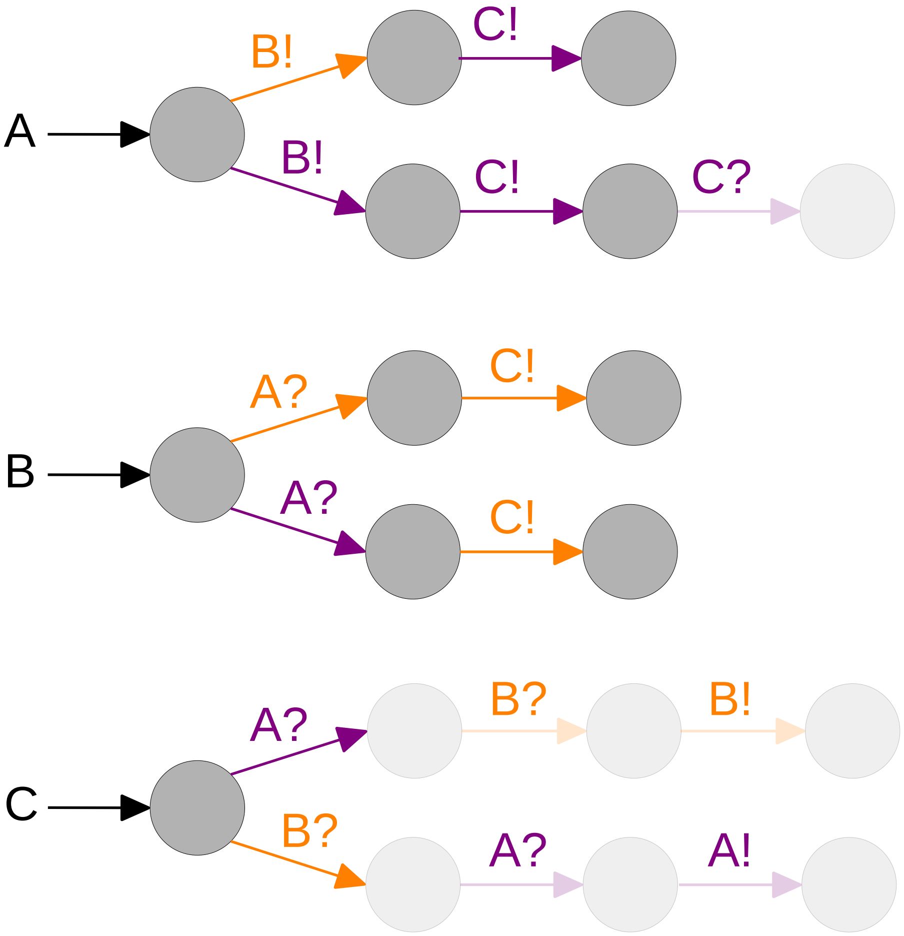


project



Which is Wrong

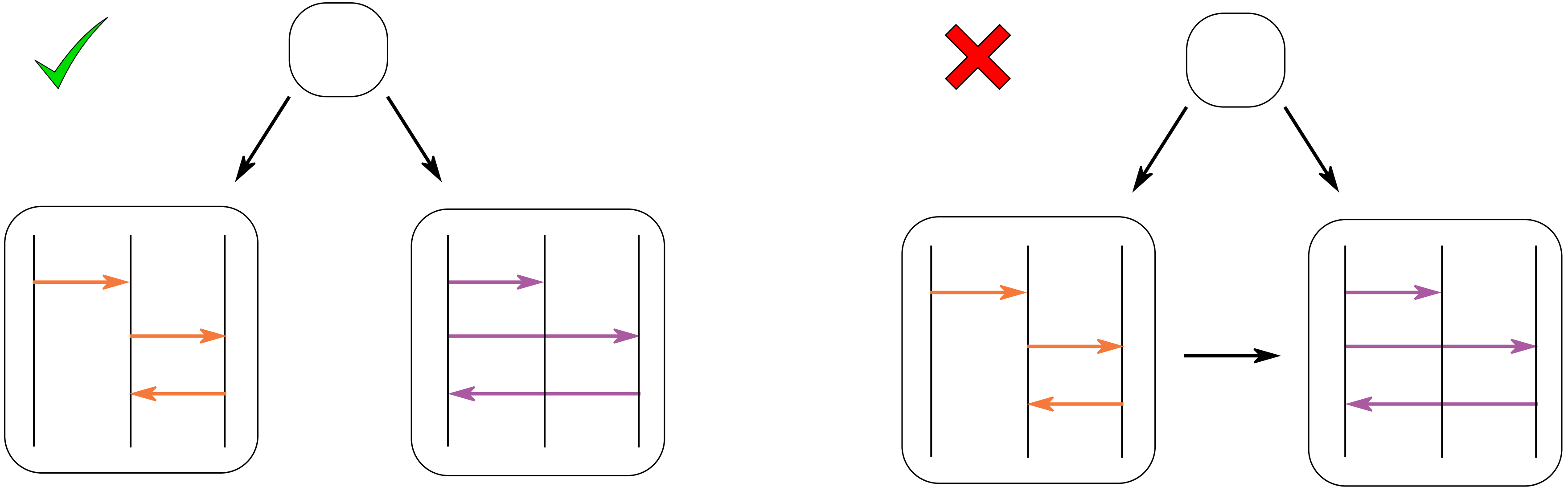




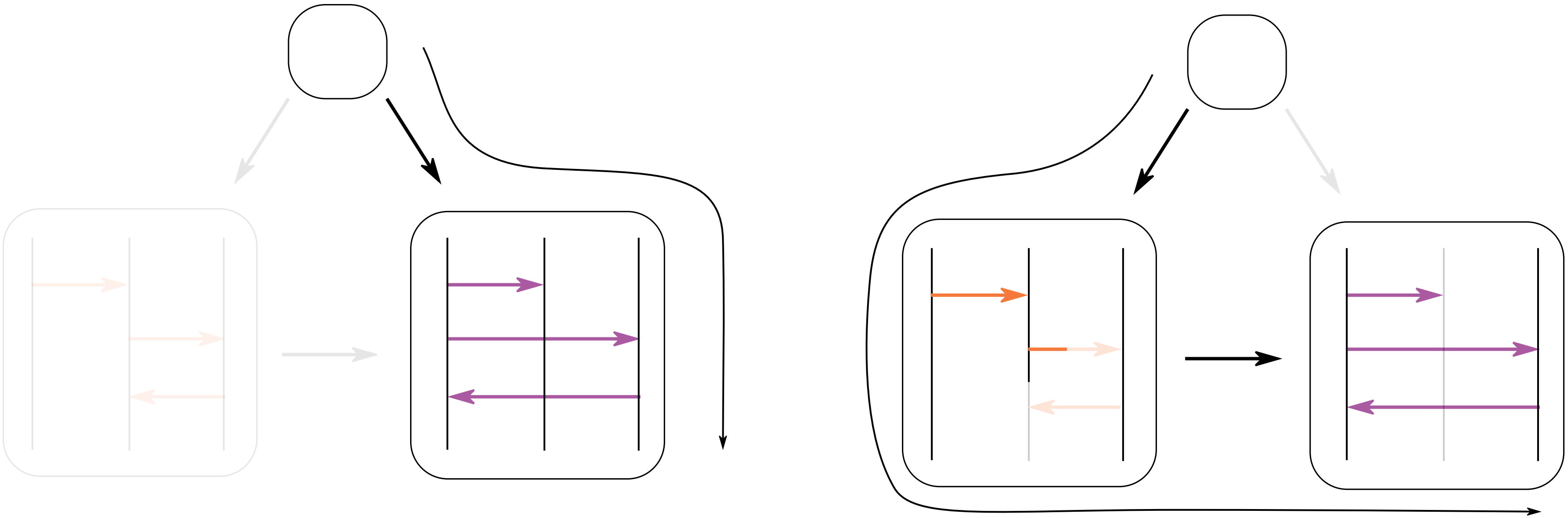
The messages from A,B are independent.
C cannot rely on their ordering.

No Compositionality

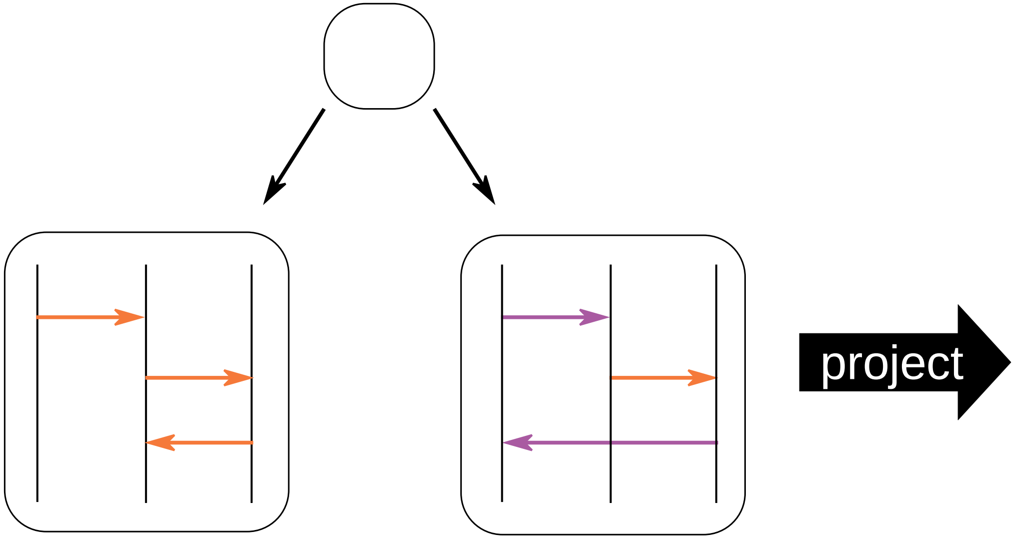
Composing two correct subprotocols can result in an incorrect one...

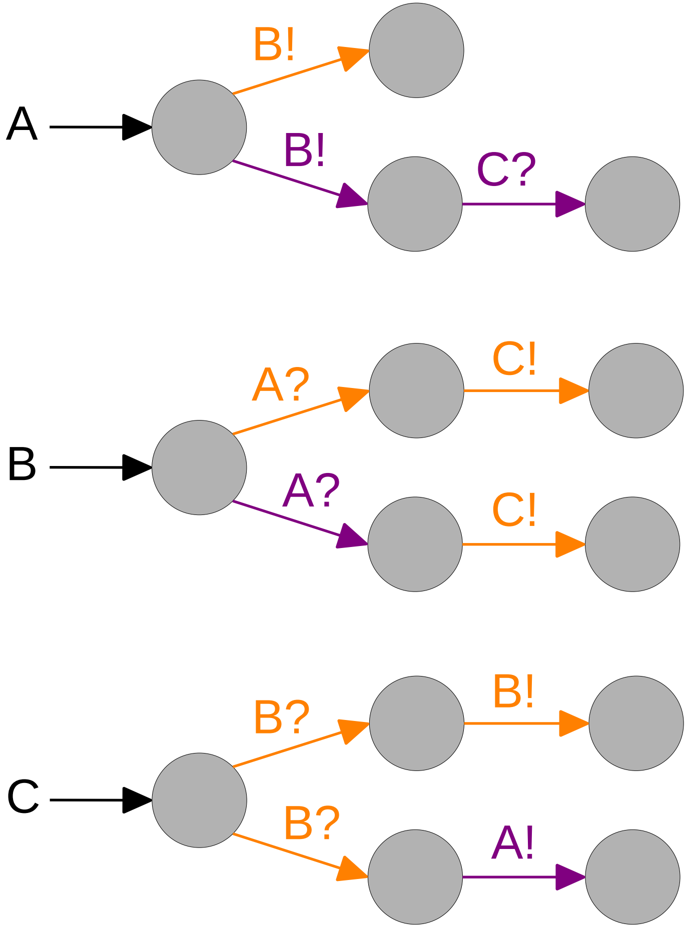


Counterexample



Determinism does not carry over

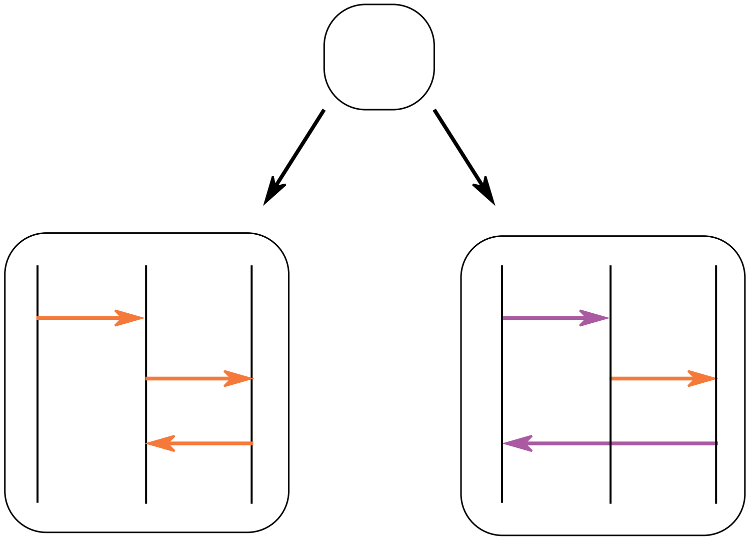


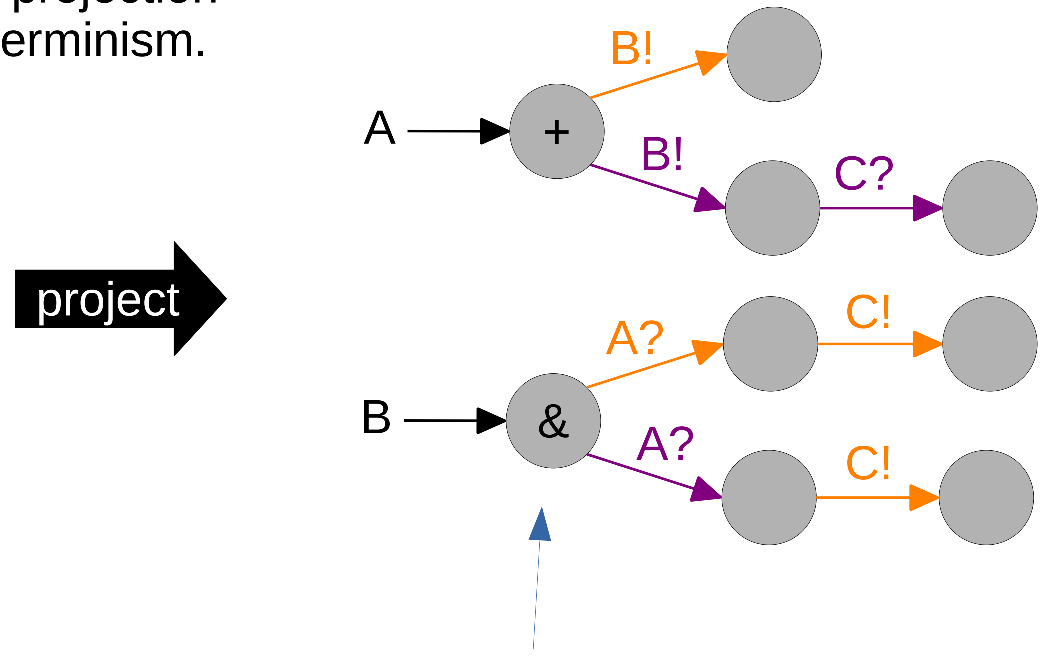


A makes the choice,
B is deterministic,
but C is not...

MST in a Nutshell (1)

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

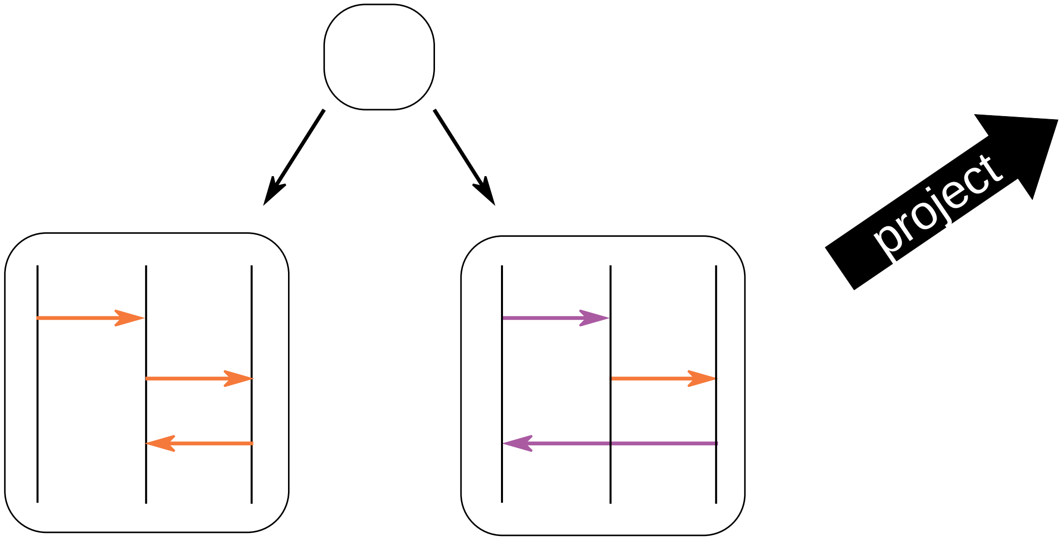




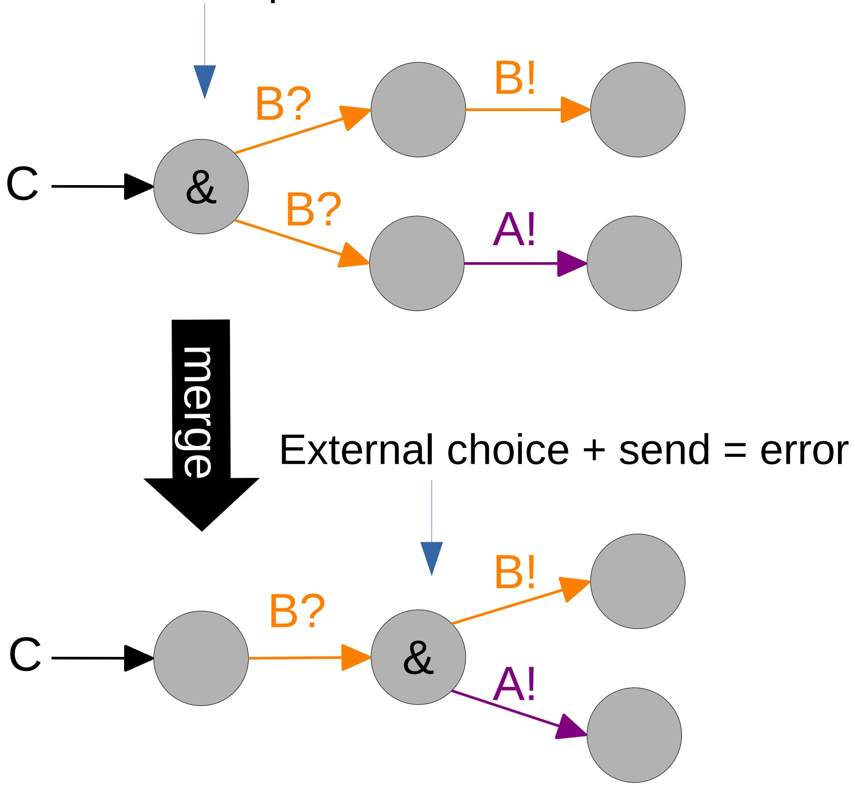
Keep track of internal/external choice

MST in a Nutshell (2)

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

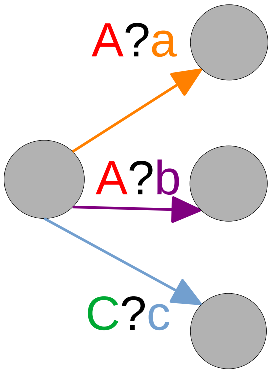


Same message on both side
Postpone choice resolution



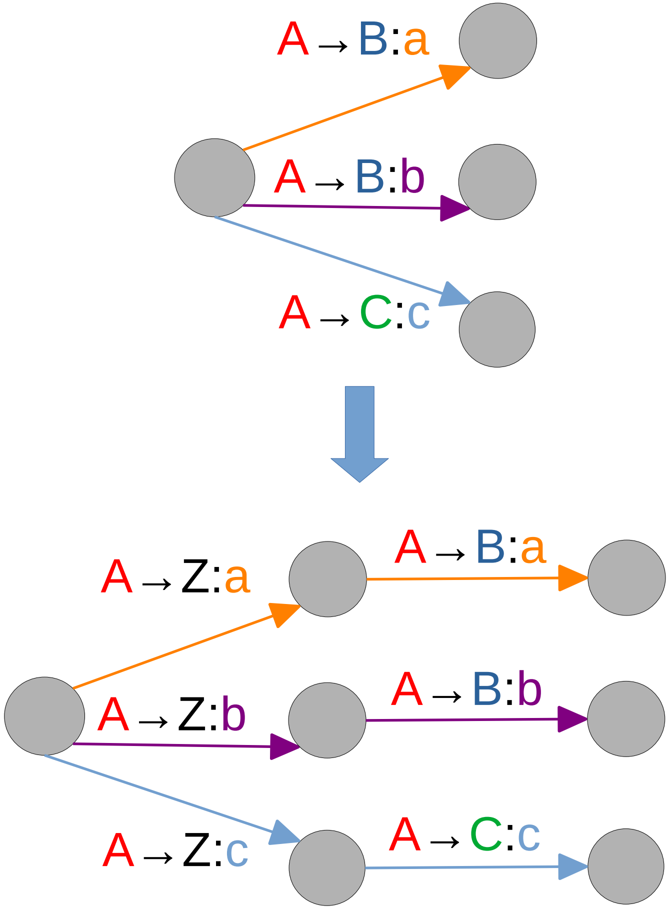
External choice + send = error

Generalized Choice

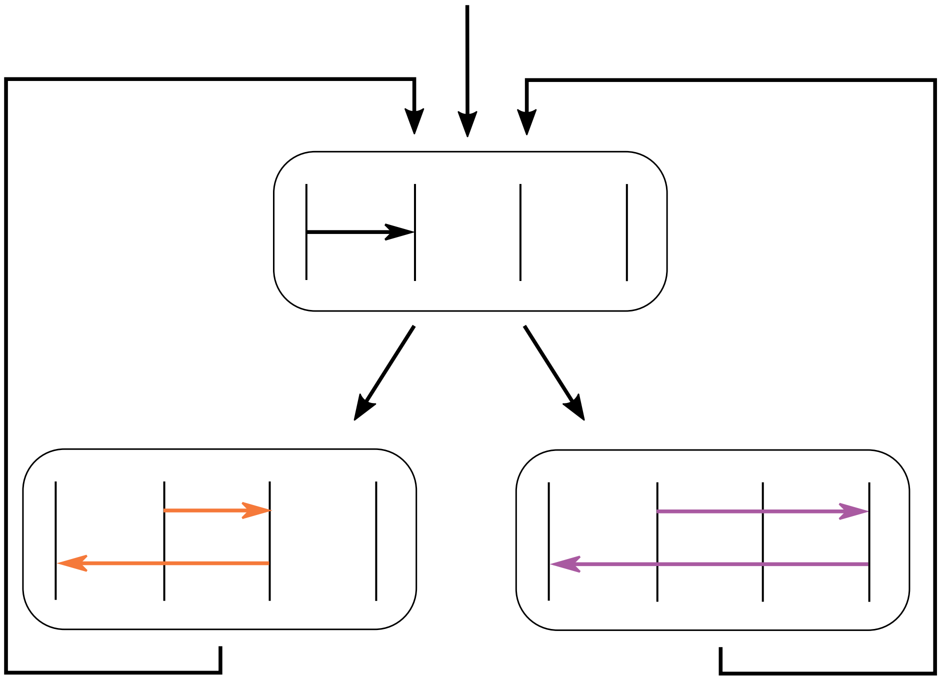


Local generalize receive that matters.

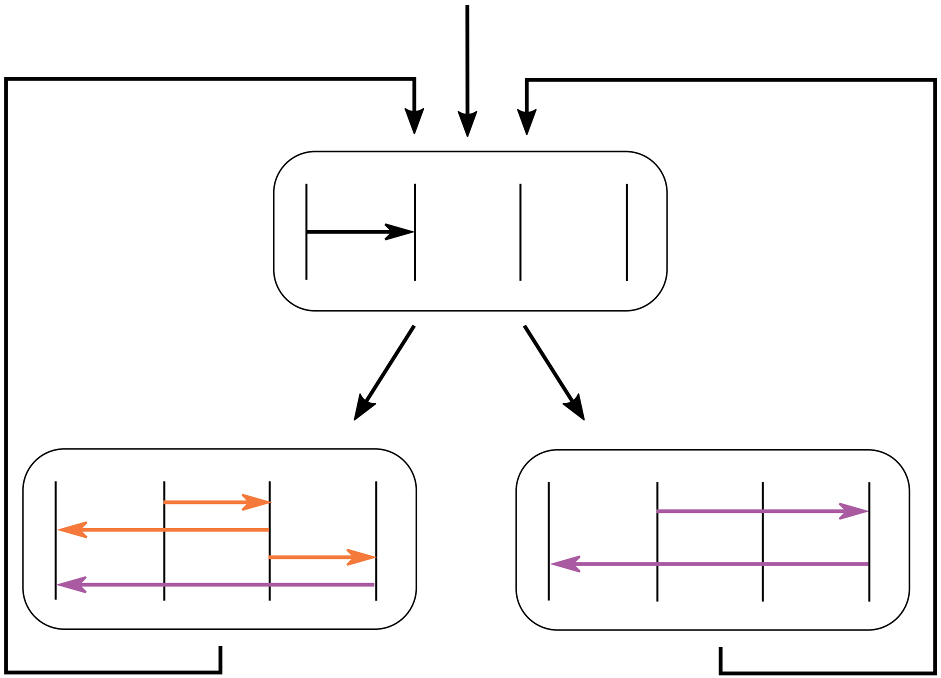
Global and sending can be simulated with an extra process.



Partial Order Across Channels

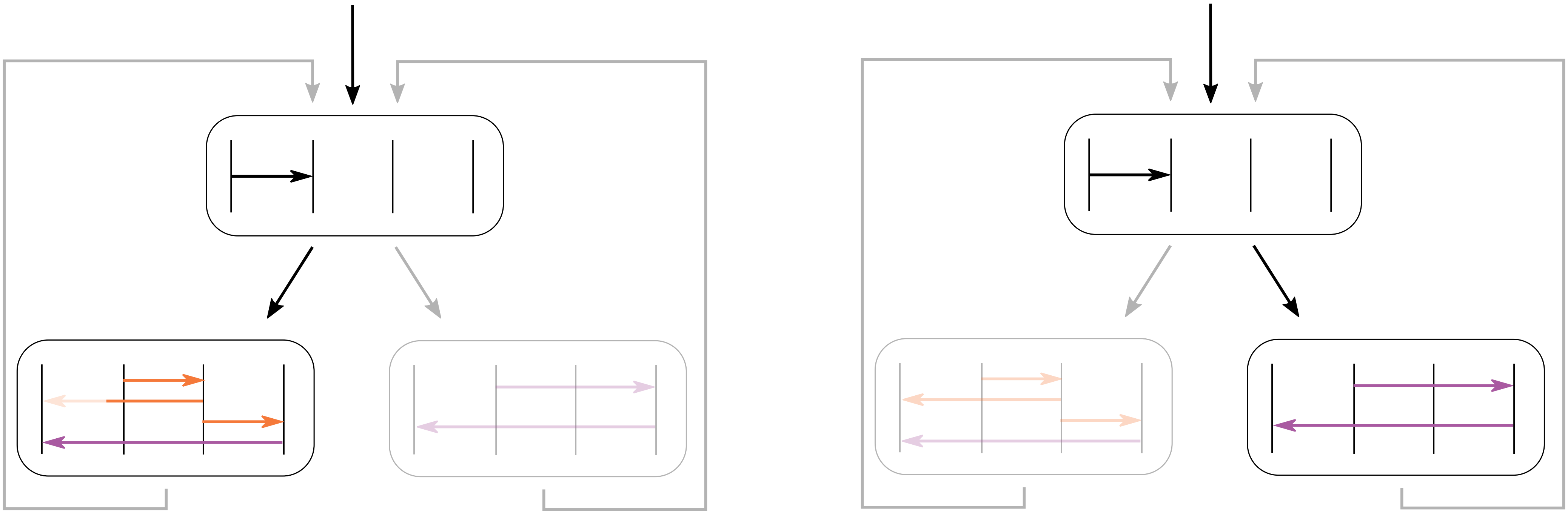


Well-formed



Wrong

Wrong Example



Projection and Merge

- Projection is similar to MST
- Merge checks for potential confusion

Available messages

$$\langle\ \&_{i\in I}\ \mathtt{q}_i?m_i.AL_{1,i}\ \ , Msg_1\rangle \sqcap \langle\ \&_{i\in J}\ \mathtt{q}_i?m_i.AL_{2,i}\ , Msg_2\rangle\ \ =$$

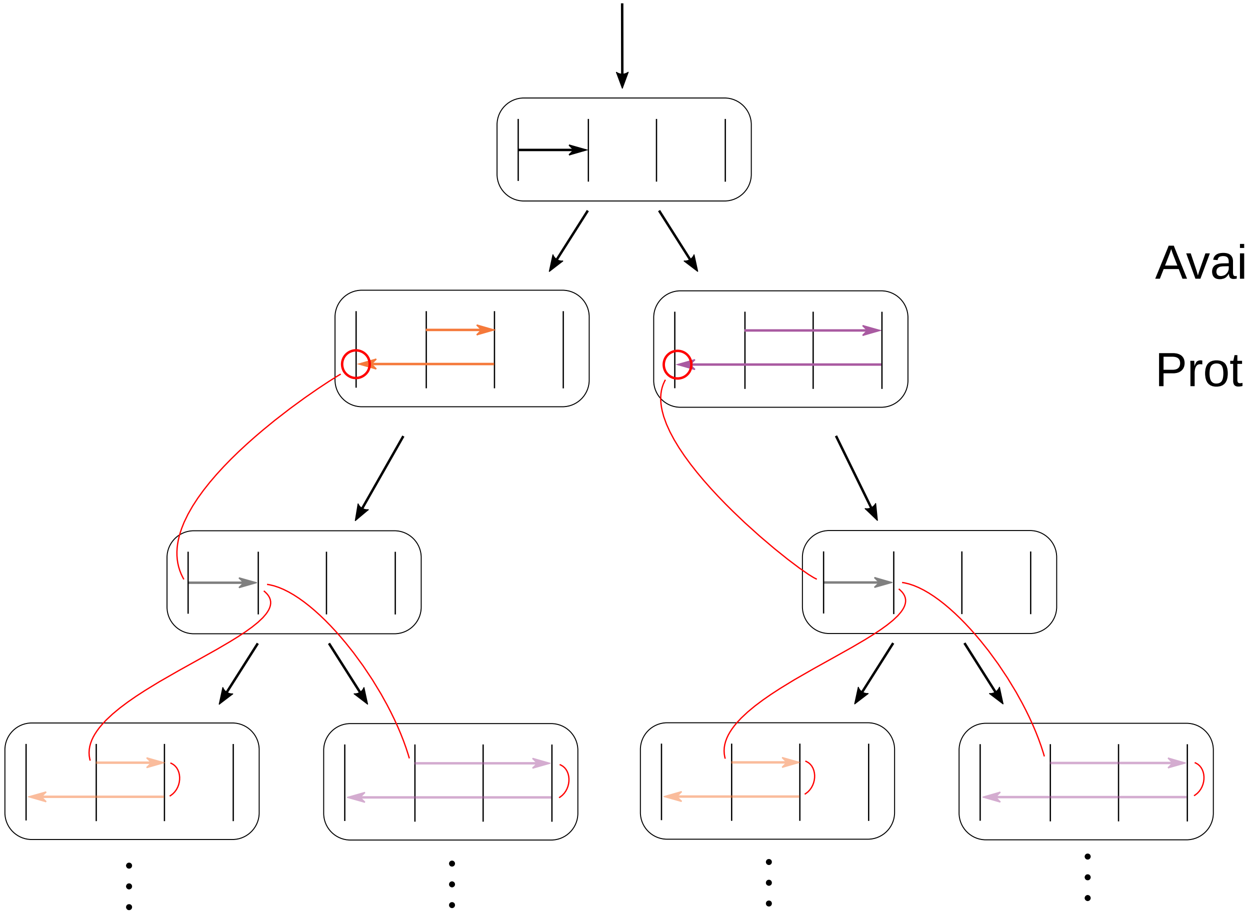
$$\begin{array}{lll} \&_{i\in I\setminus J}\ \mathtt{q}_i?m_i.AL_{1,i} & \& & \\ \&_{i\in I\cap J}\ \mathtt{q}_i?m_i.(AL_{1,i}\sqcap AL_{2,i}) & \& & \textit{if} \begin{cases} \forall i\in I\setminus J.\,\mathtt{r}\lhd\mathtt{q}_i?m_i\notin Msg_2, \\ \forall i\in J\setminus I.\,\mathtt{r}\lhd\mathtt{q}_i?m_i\notin Msg_1 \end{cases} \\ \&_{i\in J\setminus I}\ \mathtt{q}_i?m_i.AL_{2,i} & & \end{array}$$

No confusion

Available Messages

- 1st message in the channels (FIFO)
- Messages indep. from the receptions to merge
- Effectively computable (unfold recursion once)

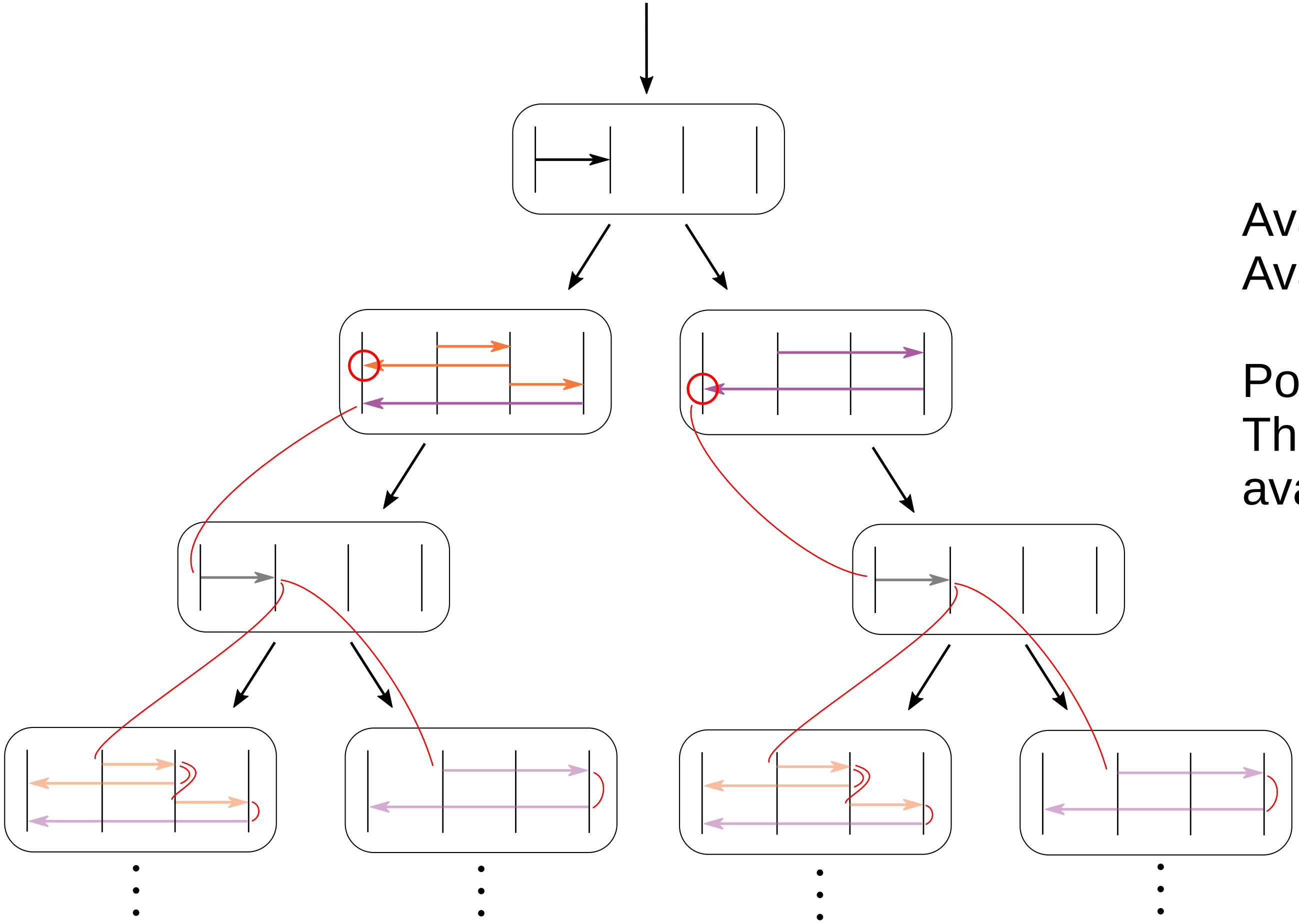
Computing Available Messages 1



Available messages are ∅.

Protocol is one long causal chain.

Computing Available Messages 2



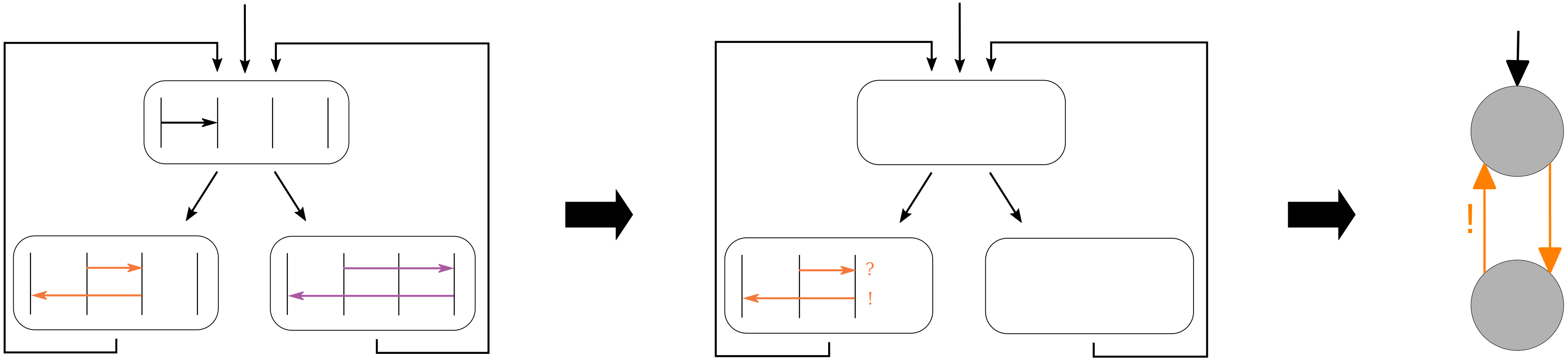
Available left branch: {←}
Available right branch: ∅

Potential confusion:
The right receive ← but it is also available on the left branch.

Empty Paths (Loops) Elimination

Each worker only appears in one branch.

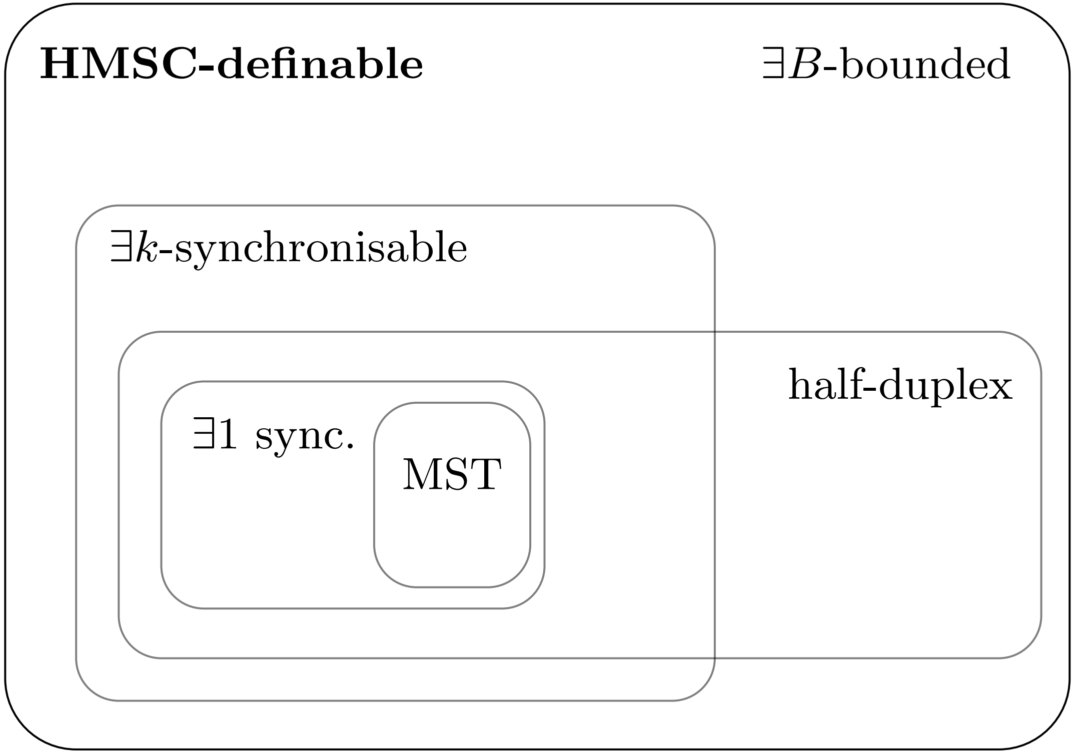
More general projection that allows loops
where a process is not involved.



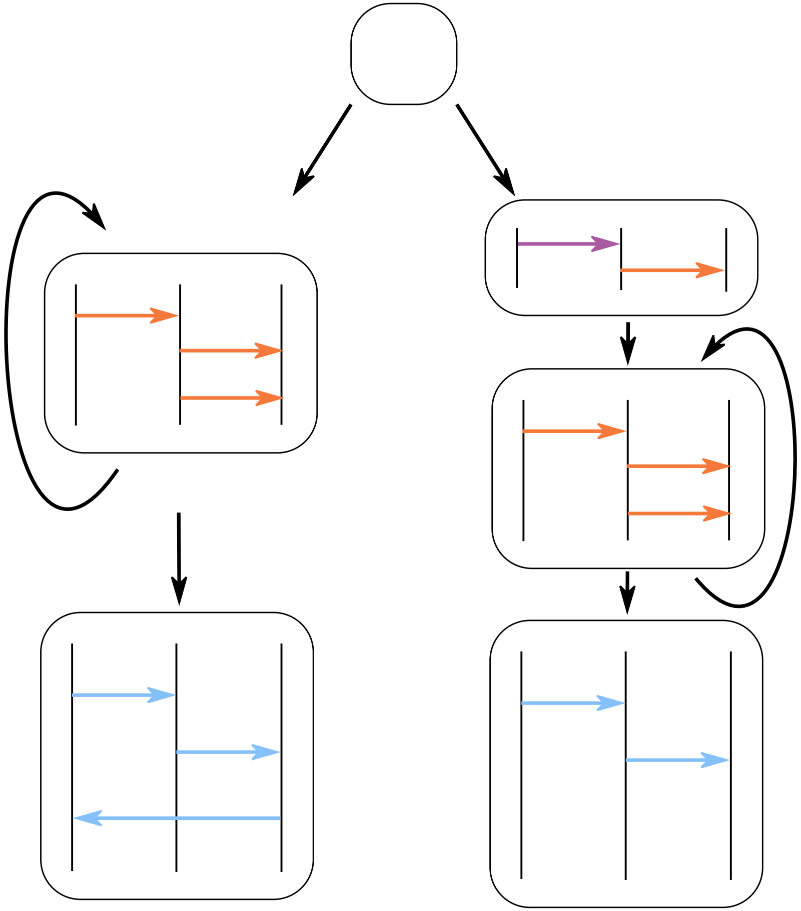
Future Work

- Understanding MST languages
- Completeness and loops
- Reordering of independent receive
- Subtyping

Understanding MST Languages



Completeness and Loops

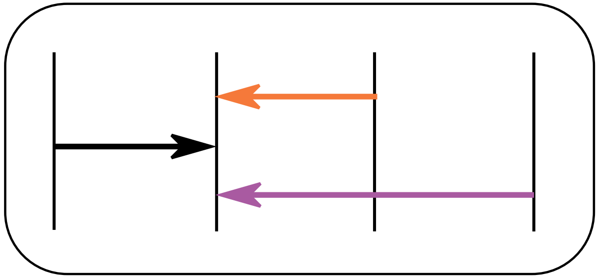


Implementable but rejected by MST.

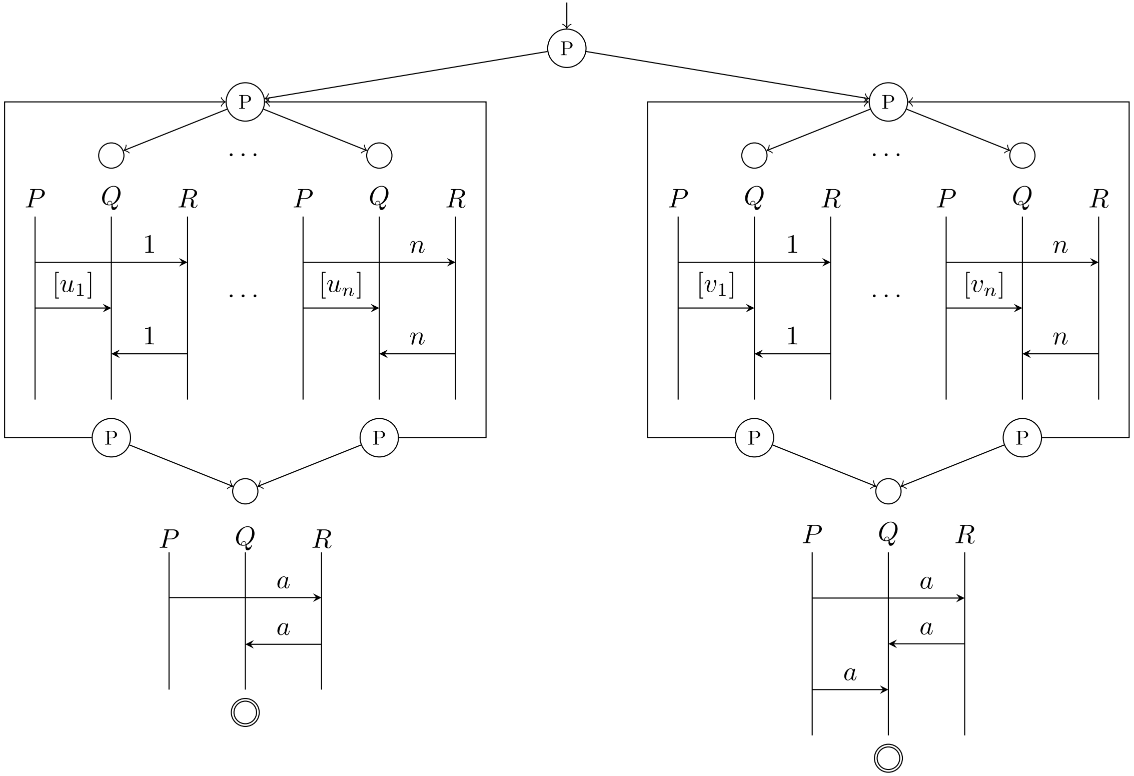
directed choice
∃1-bounded
∃1-sync
half-duplex

Intra-Process Reordering

- Somewhat similar to precise subtyping
- Motivation: performance, process independent messages in their order of arrival



Unfolding with Reordering ...



PCP encoding

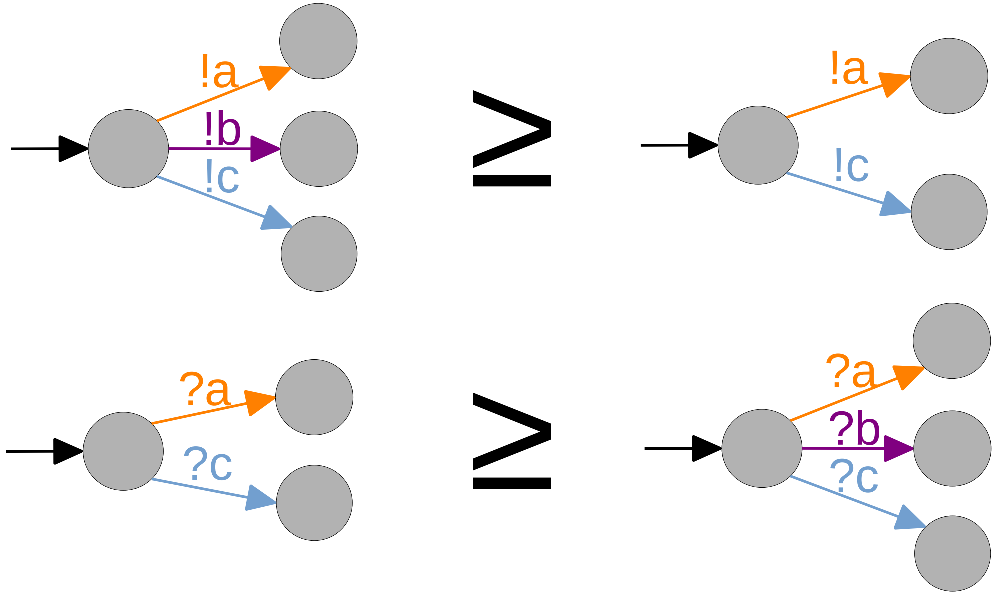
… is hard.

Subtyping

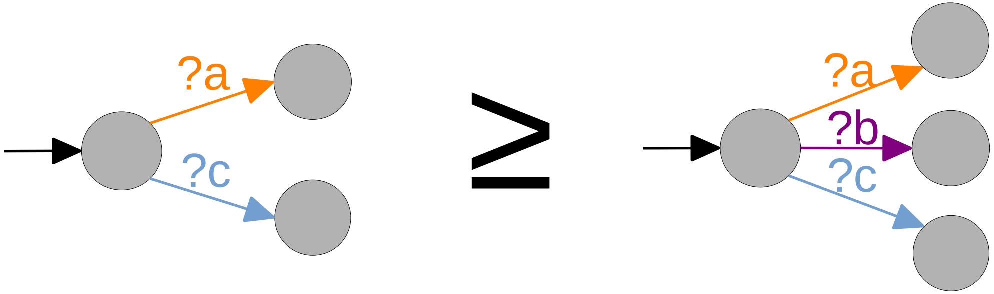
What we expect for “typical” subtyping rules

Send

Receive



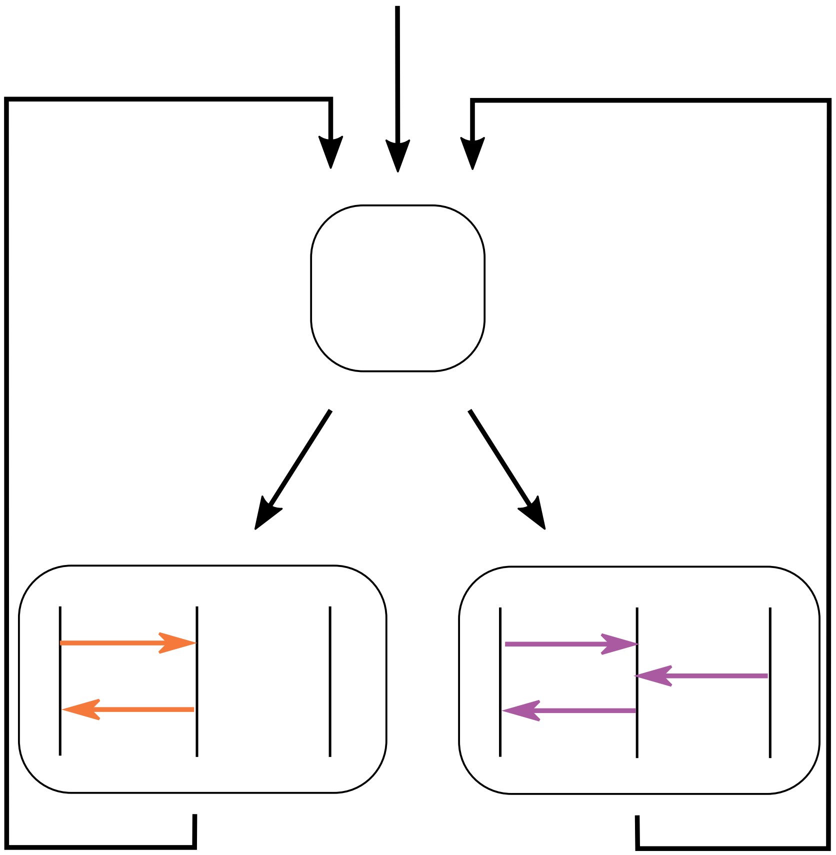
Subtyping: Receive

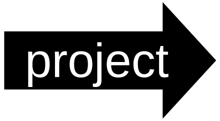


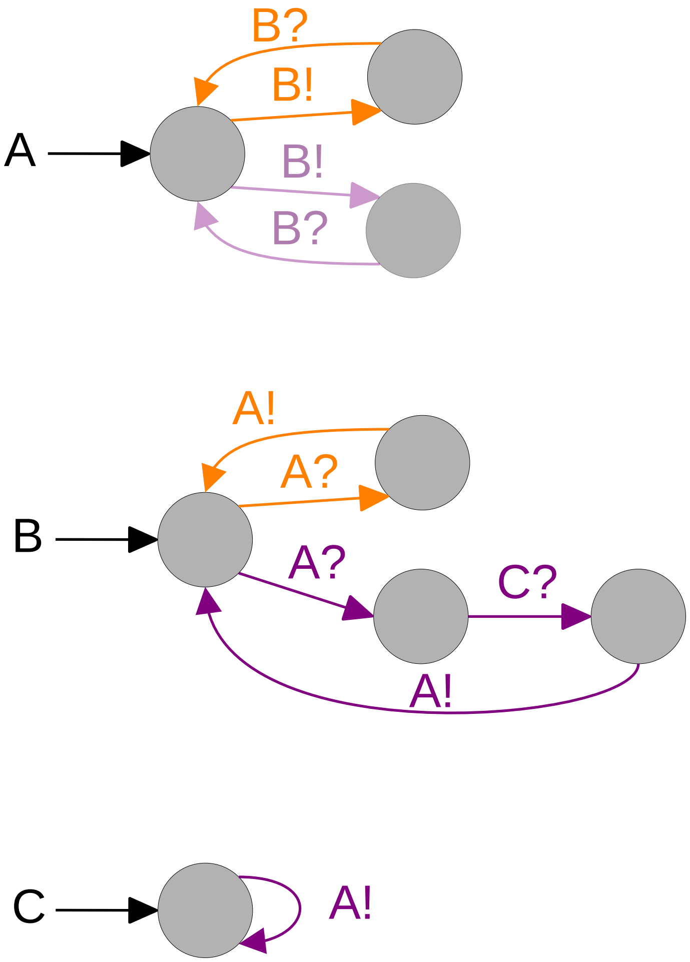
If ?b is an available messages.

Subtyping is not local but needs global information.

Subtyping: Send

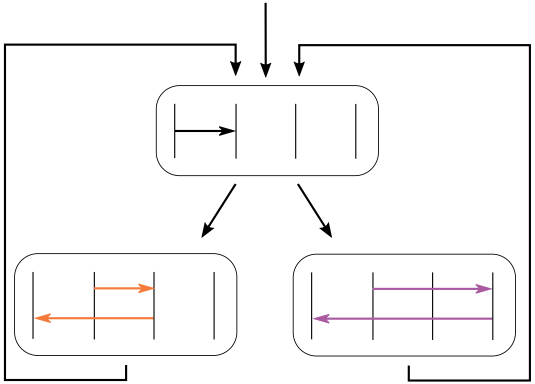






Conclusion

- Understanding how to specify communication protocols
- Finding the balance between expressiveness and complexity
- Generalized choice with projection that computes the available messages.



https://gitlab.mpi-sws.org/fstutz/async-mpst-gen-choice