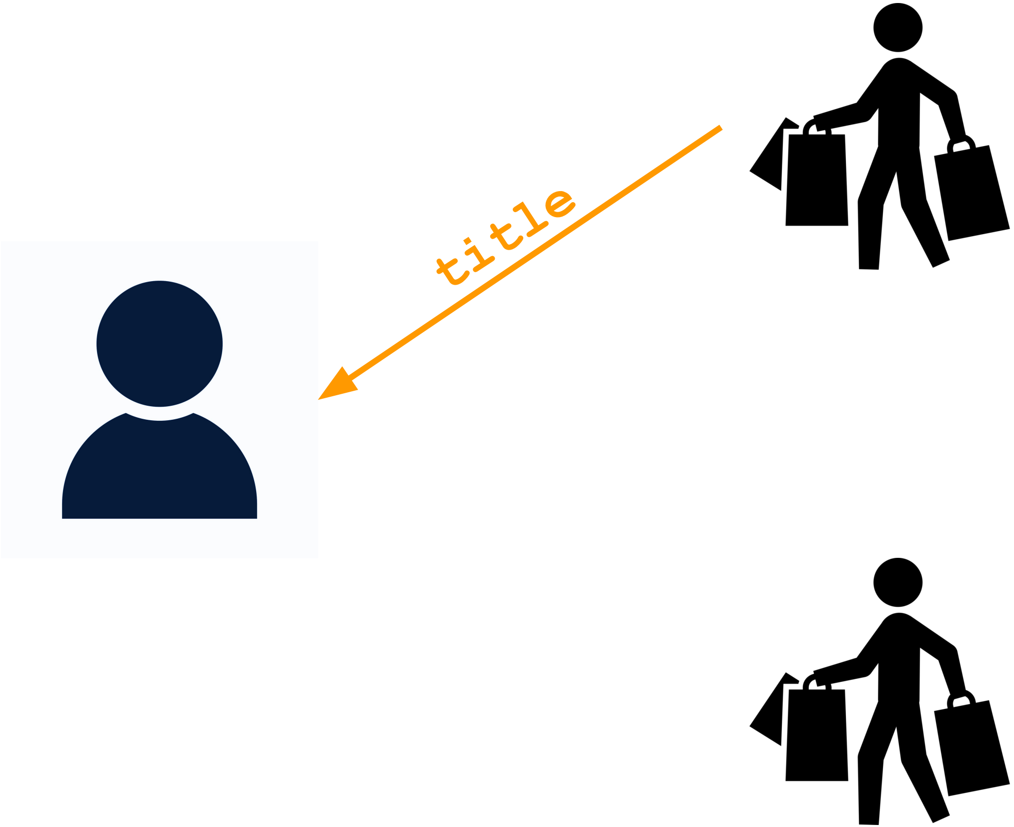
Deciding Subtyping for Asynchronous Multiparty Sessions

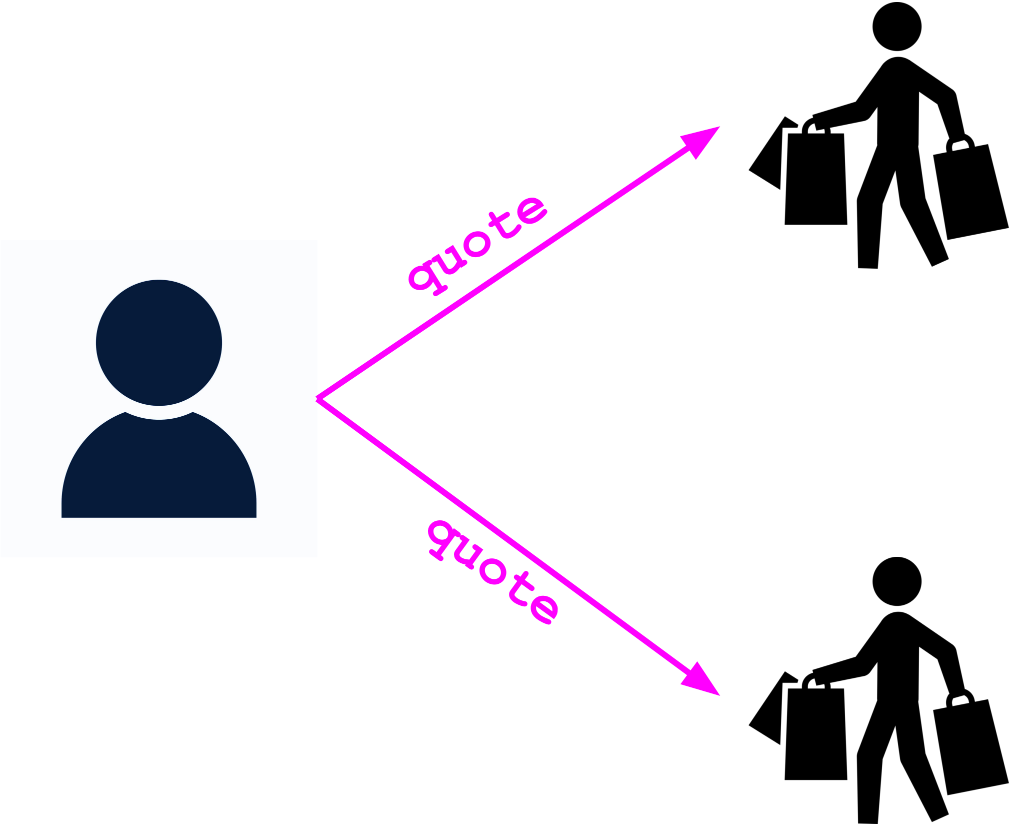
Elaine Li Felix Stutz Thomas Wies

NYU NYU

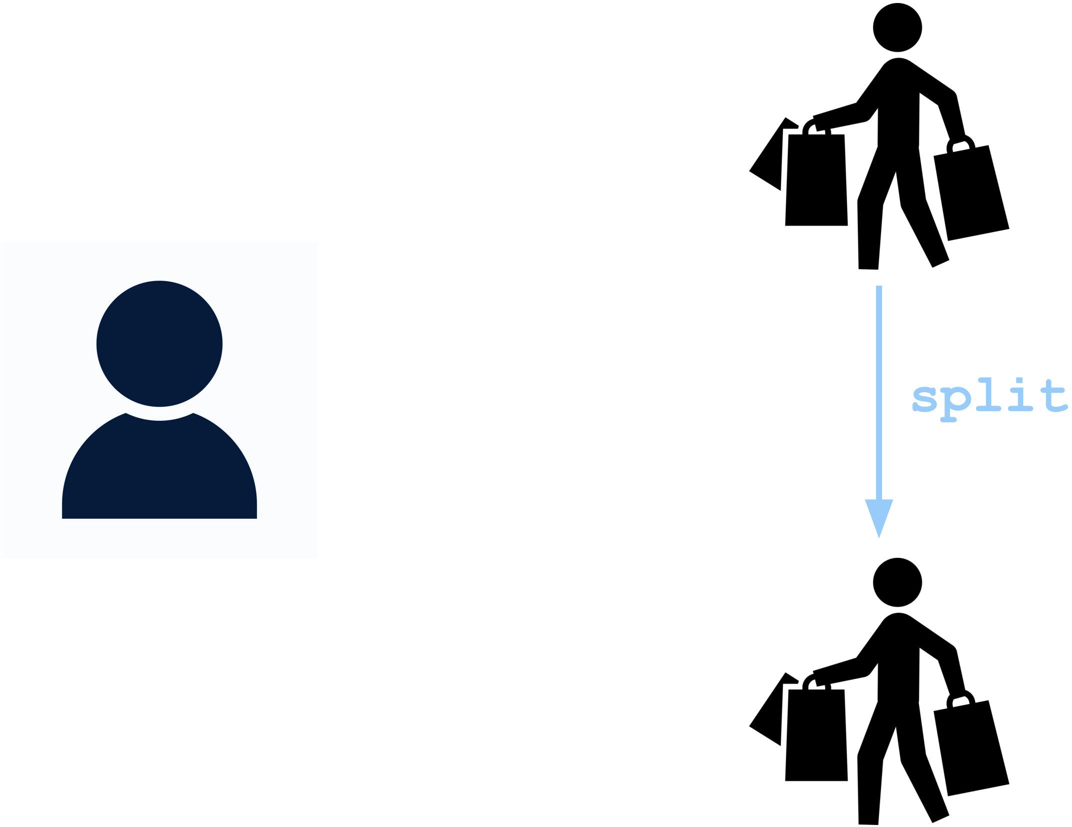
Multiparty session types: two-buyer protocol



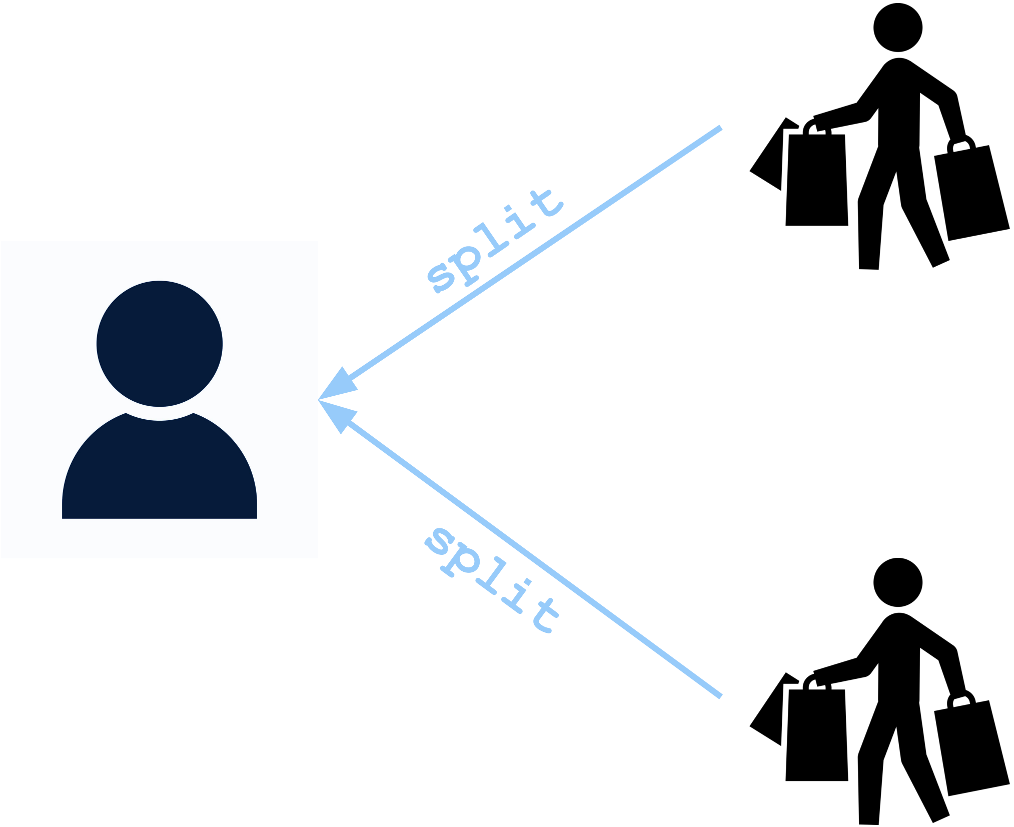
Multiparty session types: two-buyer protocol



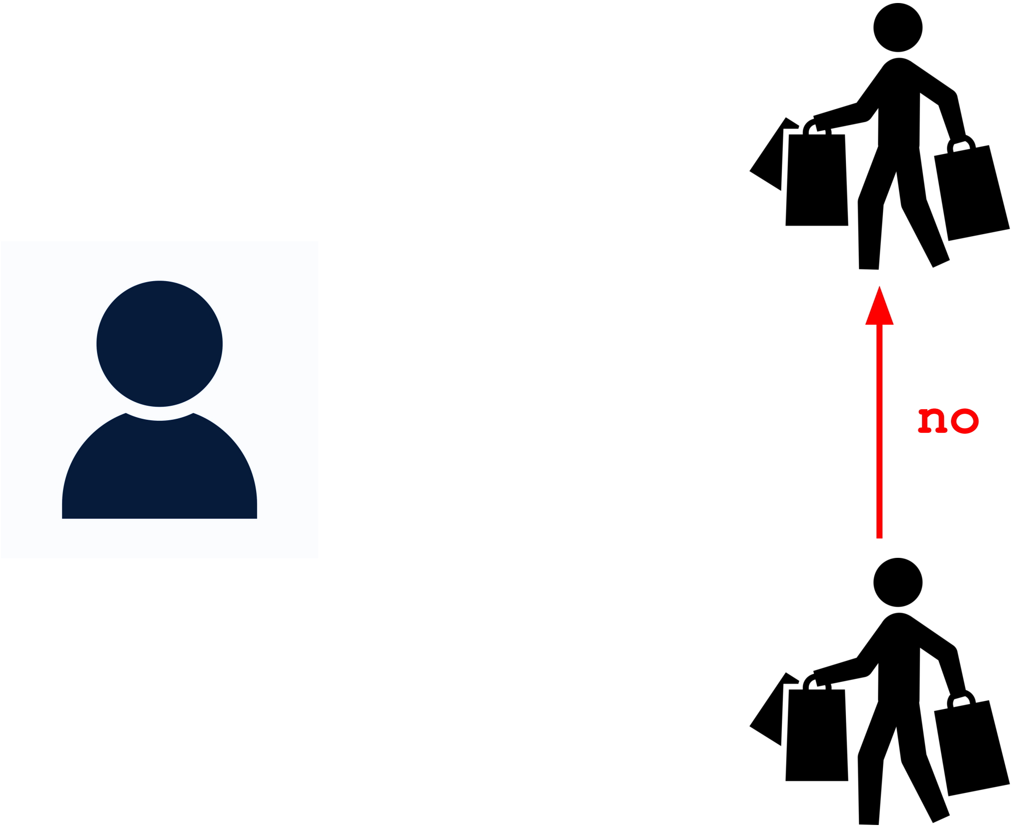
Multiparty session types: two-buyer protocol



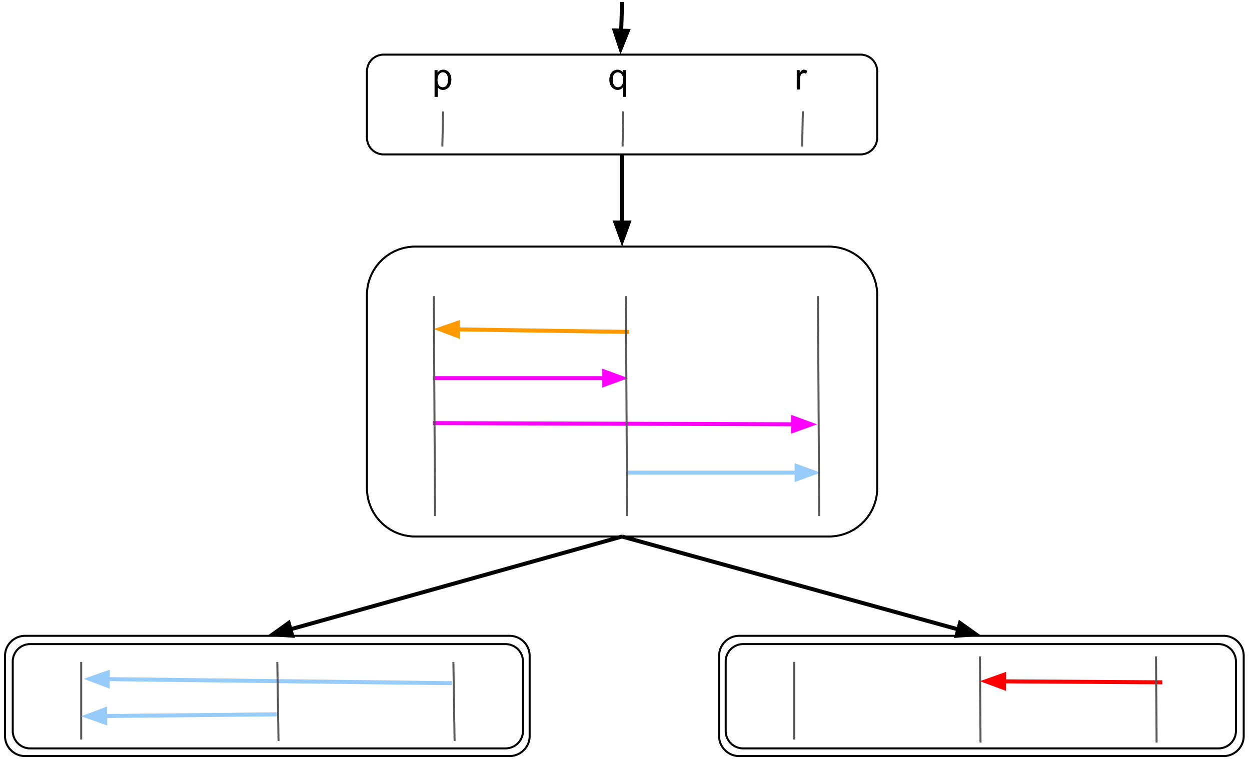
Multiparty session types: two-buyer protocol



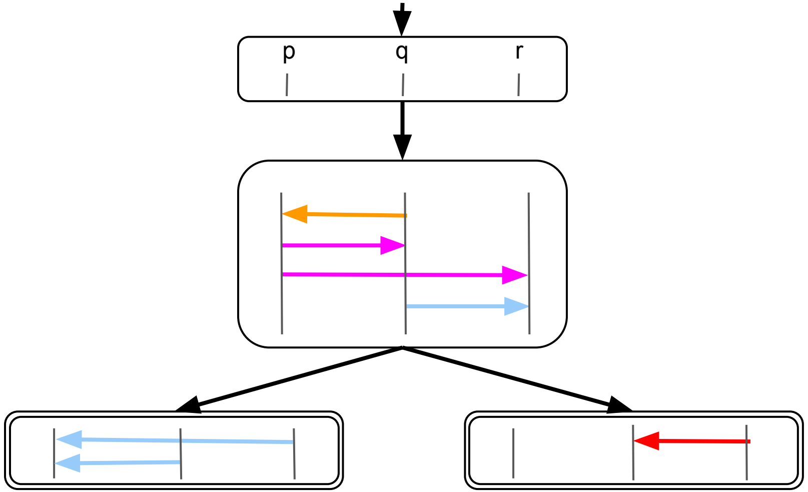
Multiparty session types: two-buyer protocol



Multiparty session types

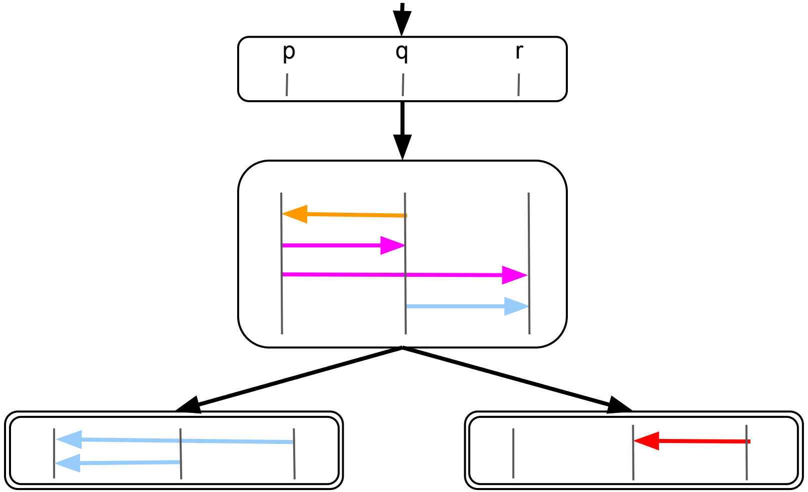


MST semantics



Synchronous $\mathtt{q}{\rightarrow}\mathtt{p}{:}o \cdot \mathtt{p}{\rightarrow}\mathtt{q}{:}m \cdot \mathtt{p}{\rightarrow}\mathtt{r}{:}m \cdot \mathtt{q}{\rightarrow}\mathtt{r}{:}b \ldots$

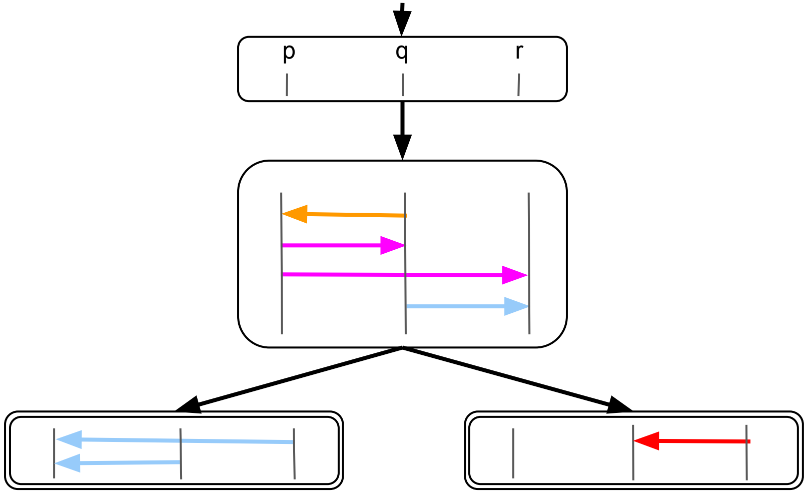
MST semantics



Synchronous $\mathtt{q}\to\mathtt{p}:o \cdot \mathtt{p}\to\mathtt{q}:m \cdot \mathtt{p}\to\mathtt{r}:m \cdot \mathtt{q}\to\mathtt{r}:b \ldots$

Asynchronous $\mathtt{q} \triangleright \mathtt{p}!o \cdot \mathtt{p} \triangleleft \mathtt{q}?o \cdot \mathtt{p} \triangleright \mathtt{q}!m \cdot \mathtt{q} \triangleleft \mathtt{p}?m \cdot \mathtt{p} \triangleright \mathtt{r}!m \cdot \mathtt{r} \triangleleft \mathtt{p}?m \cdot \mathtt{q} \triangleright \mathtt{r}!b \cdot \mathtt{r} \triangleleft \mathtt{q}?b \ldots$

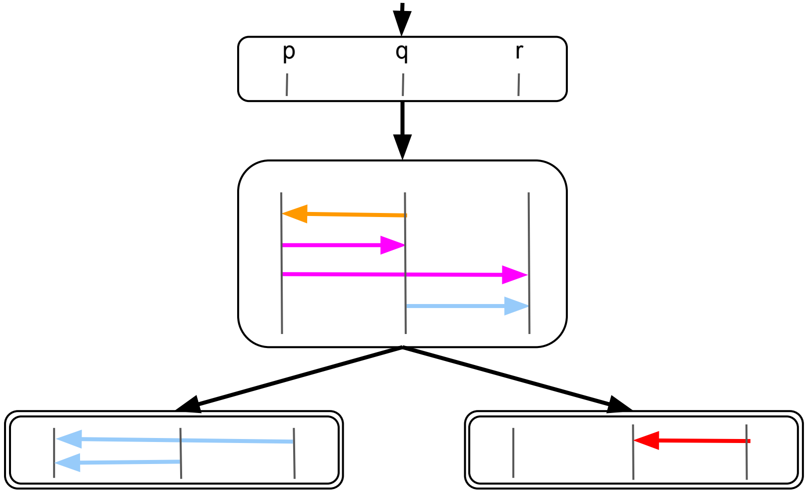
MST semantics



Synchronous $\mathtt{q} \to \mathtt{p} : o \cdot \mathtt{p} \to \mathtt{q} : m \cdot \mathtt{p} \to \mathtt{r} : m \cdot \mathtt{q} \to \mathtt{r} : b \ldots$

Asynchronous $\mathtt{q} \triangleright \mathtt{p}!o \cdot \mathtt{p} \triangleleft \mathtt{q}?o \cdot \mathtt{p} \triangleright \mathtt{q}!m \cdot \mathtt{q} \triangleleft \mathtt{p}?m \cdot \mathtt{p} \triangleright \mathtt{r}!m \cdot \mathtt{r} \triangleleft \mathtt{p}?m \cdot \mathtt{q} \triangleright \mathtt{r}!b \cdot \mathtt{r} \triangleleft \mathtt{q}?b \ldots$

MST semantics



Synchronous $\mathtt{q} \rightarrow \mathtt{p} : o \cdot \mathtt{p} \rightarrow \mathtt{q} : m \cdot \mathtt{p} \rightarrow \mathtt{r} : m \cdot \mathtt{q} \rightarrow \mathtt{r} : b \ldots$

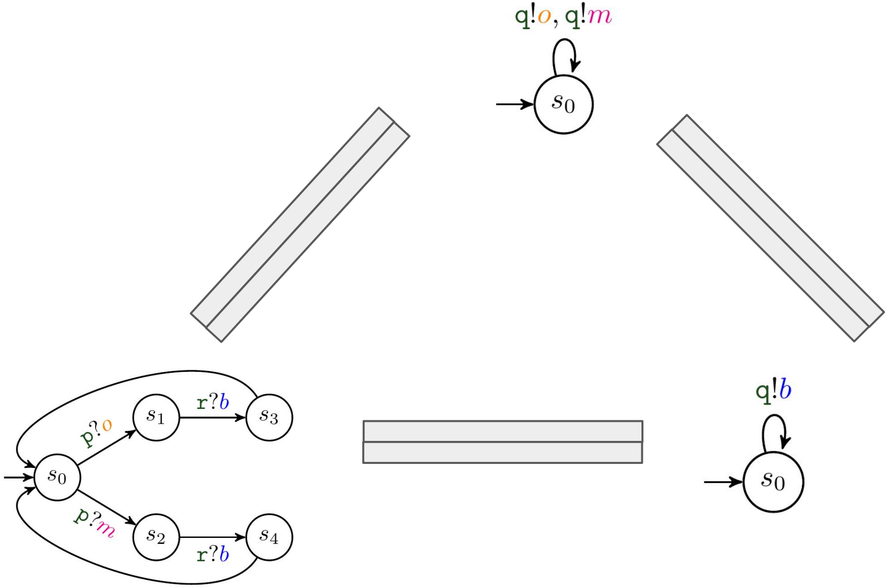
Asynchronous $\mathtt{q} \triangleright \mathtt{p}!o \cdot \mathtt{p} \triangleleft \mathtt{q}?o \cdot \mathtt{p} \triangleright \mathtt{q}!m \cdot \mathtt{q} \triangleleft \mathtt{p}?m \cdot \mathtt{p} \triangleright \mathtt{r}!m \cdot \mathtt{r} \triangleleft \mathtt{p}?m \cdot \mathtt{q} \triangleright \mathtt{r}!b \cdot \mathtt{r} \triangleleft \mathtt{q}?b \ldots$

Interleaving $\mathtt{q} \triangleright \mathtt{p}!o \cdot \mathtt{p} \triangleleft \mathtt{q}?o \cdot \mathtt{p} \triangleright \mathtt{q}!m \cdot \mathtt{p} \triangleright \mathtt{r}!m \cdot \mathtt{q} \triangleleft \mathtt{p}?m \cdot \mathtt{r} \triangleleft \mathtt{p}?m \cdot \mathtt{q} \triangleright \mathtt{r}!b \cdot \mathtt{r} \triangleleft \mathtt{q}?b \ldots$

MST implementations

Communicating State Machines (CSM) [Brand and Zafiropulo, JACM'83]

CSM configurations **(s, ξ)** contain **global state** and **channel state**

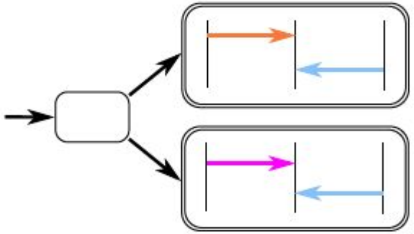


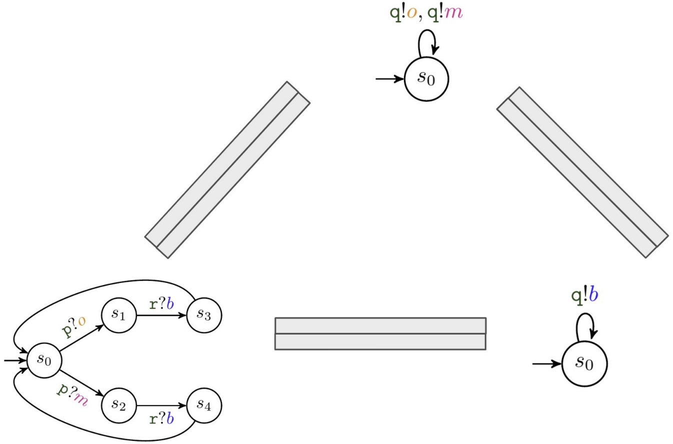
[p := s_0,
q := s_4,
r := s_0]
[(p,q) := ε,
(p,r) := ε,
(q,r) := ε,
(q,p) := ε,
(r,p) := o,
(r,q) := ε]

$\mathtt{p} \triangleright \mathtt{q}!o \cdot \mathtt{q} \triangleleft \mathtt{p}?o \cdot \mathtt{r} \triangleright \mathtt{q}!b \cdot \mathtt{q} \triangleleft \mathtt{r}?b \cdot \mathtt{p} \triangleright \mathtt{q}!m \cdot \mathtt{q} \triangleleft \mathtt{p}?m \cdot \mathtt{r} \triangleright \mathtt{q}!b \cdot \mathtt{q} \triangleleft \mathtt{r}?b \ldots$

MST implementability

Implementability = exists a CSM, protocol fidelity + deadlock freedom

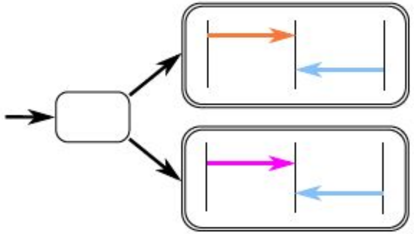




1) CSM language = global type language
2) CSM is deadlock-free

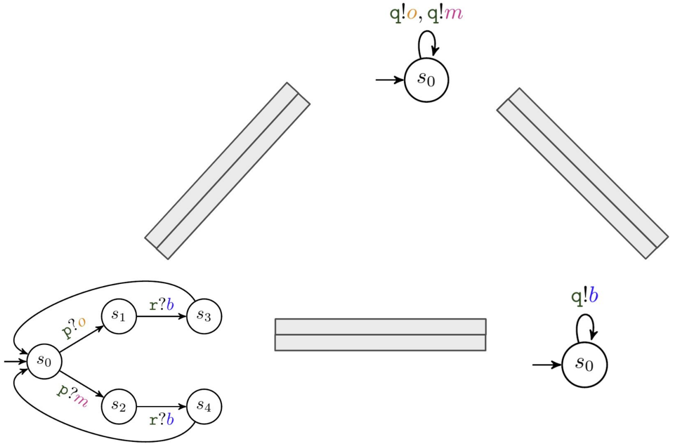
MST implementability

Implementability = exists a CSM, protocol fidelity + deadlock freedom



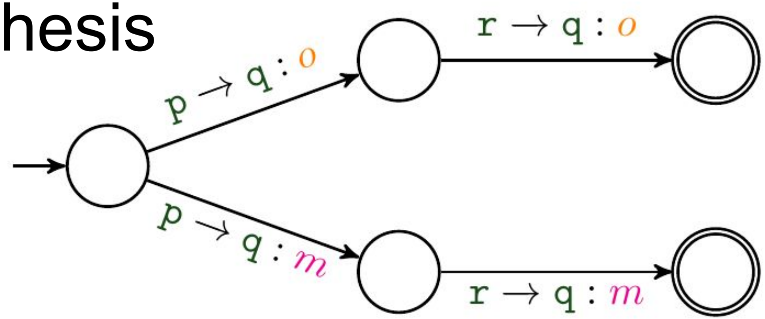
?

first sound and complete projection operator [CAV'23]

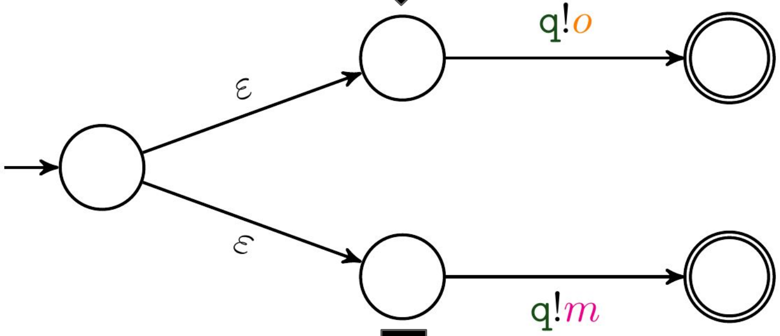


1) CSM language = global type language
2) CSM is deadlock-free

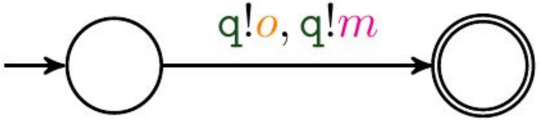
Projection: Synthesis



homomorphism for r

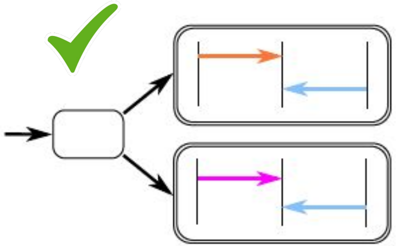


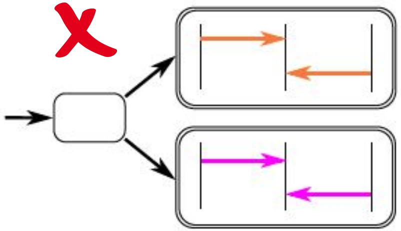
determinization



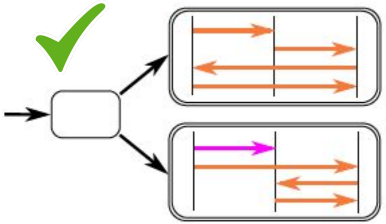
Projection: Checking Implementability

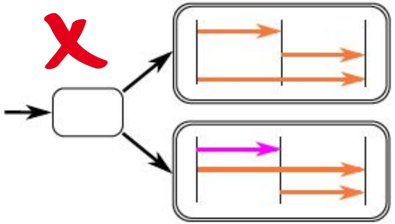
1. Send Validity: "send transitions originate from all global states"





2. Receive Validity: "receive transitions uniquely identify a global state"





MST implementability

Implementability = exists a CSM, protocol fidelity + deadlock freedom

$\mathbf{G(}\{0,1\}^*0\{0,1\}^{n-1}\mathbf{)}$ $\xrightarrow{?}$

first sound and complete projection operator [CAV'23]

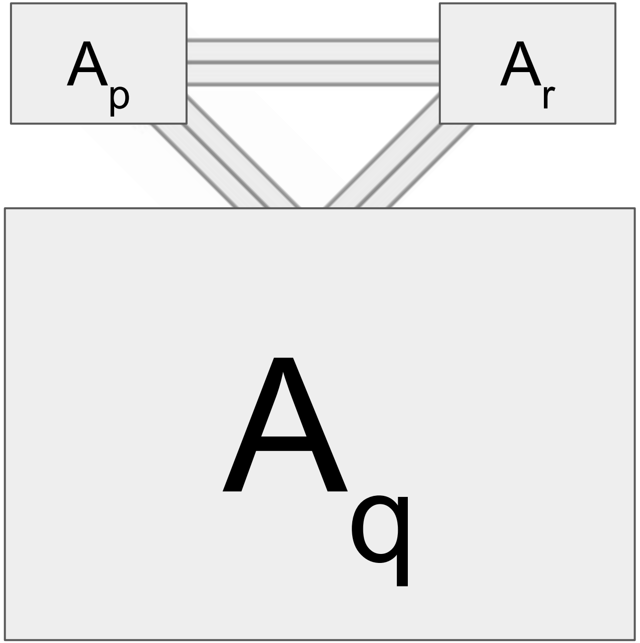
MST implementability

Implementability = exists a CSM, protocol fidelity + deadlock freedom

$\mathbf{G(}\{0,1\}^*0\{0,1\}^{n-1}\mathbf{)}$

?

first sound and complete projection operator [CAV'23]

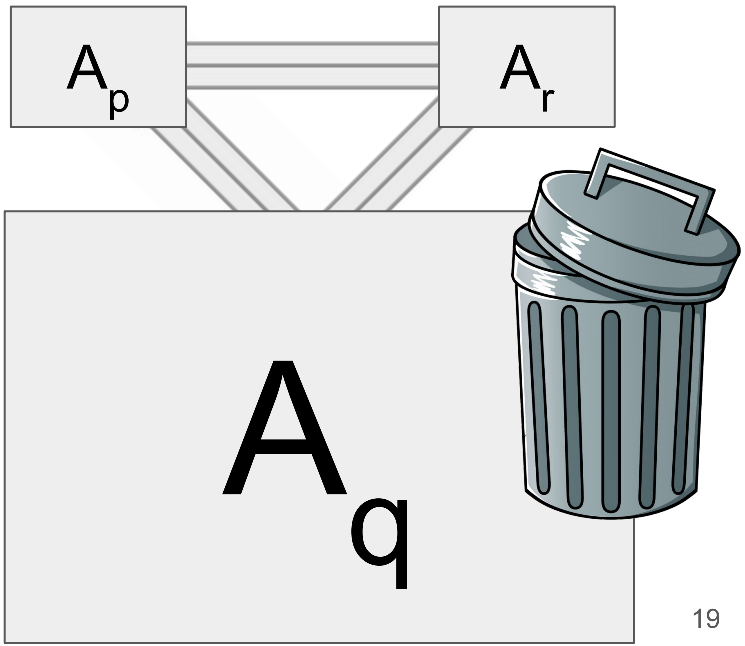


MST implementability

Implementability = exists a CSM, protocol fidelity + deadlock freedom

$$\mathbf{G}(\{0,1\}^*0\{0,1\}^{n-1}) \xrightarrow{?}$$

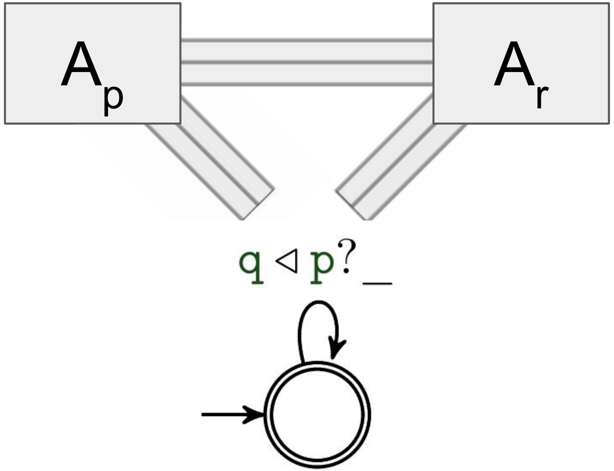
first sound and complete projection operator [CAV'23]



MST protocol verification

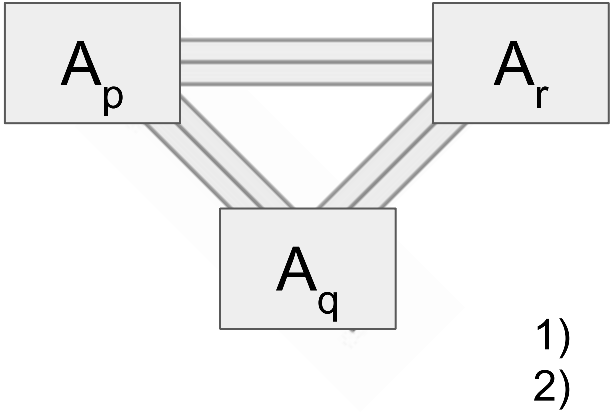
$\mathbf{G(}\{0,1\}{*}0\{0,1\}^{n-1}\mathbf{)}$

implements

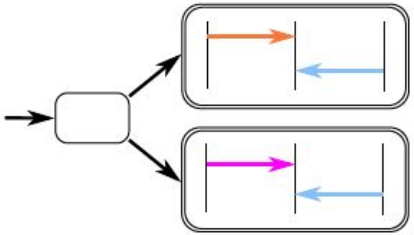


MST protocol verification

Protocol verification = *given* a CSM, protocol fidelity + deadlock freedom

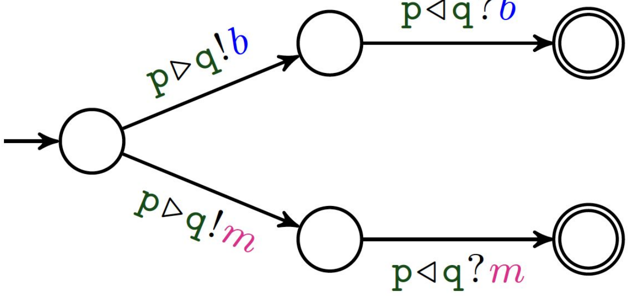


implements?

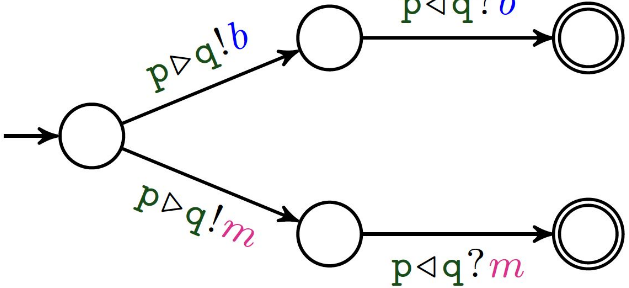


1) CSM language = global type language
2) CSM is deadlock-free

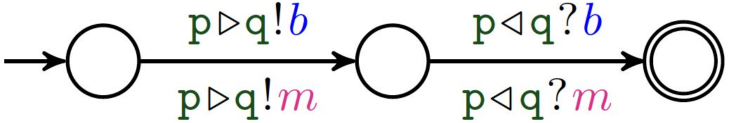
MST protocol verification



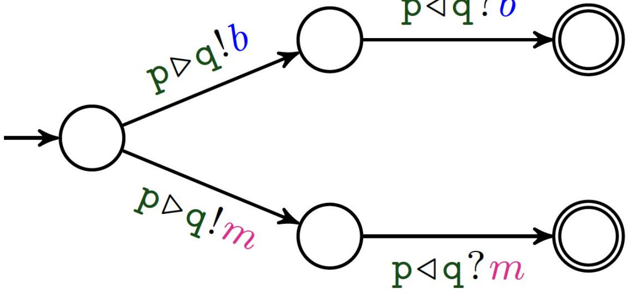
MST protocol verification



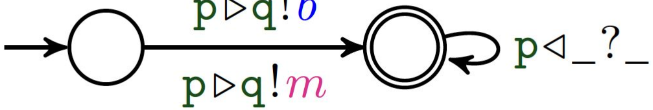
Collapsing states



MST protocol verification

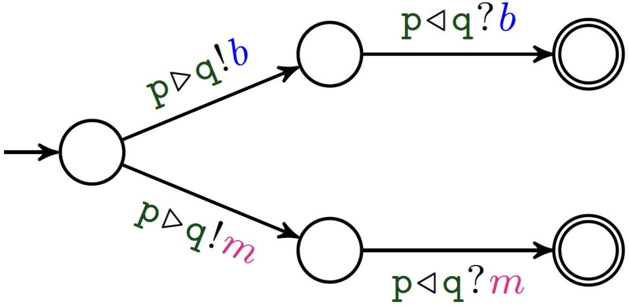


Collapsing states

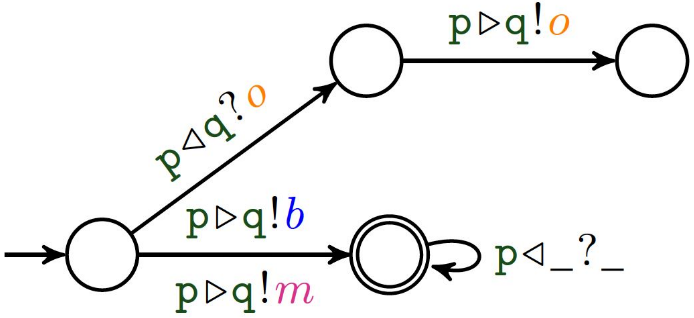


Adding receives

MST protocol verification



Collapsing states



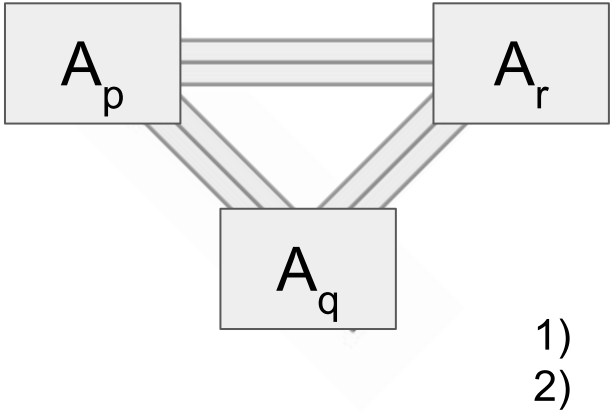
Adding unreachable states

Adding receives

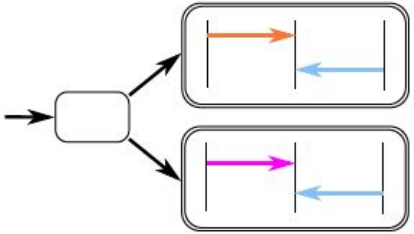
MST protocol verification

decidable in polynomial time, compositional

Protocol verification = *given* a CSM, protocol fidelity + deadlock freedom



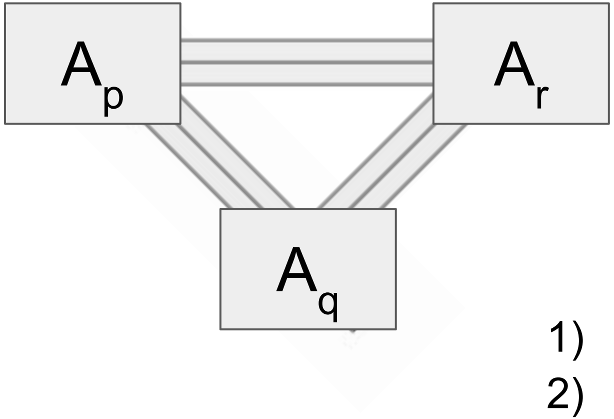
implements?



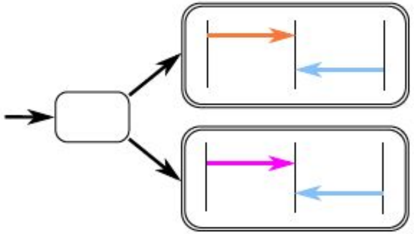
1) CSM language = global type language
2) CSM is deadlock-free

MST monolithic protocol refinement

Monolithic protocol refinement = *given* a CSM, *sub*protocol fidelity + deadlock freedom



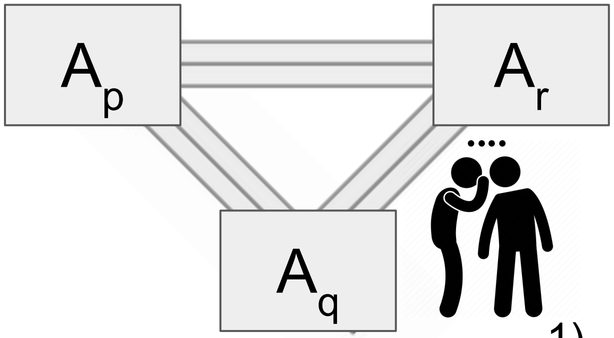
protocol refines?



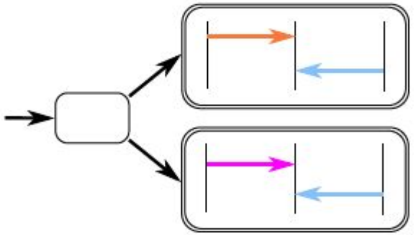
1) CSM language $\subseteq$ global type language
2) CSM is deadlock-free

MST monolithic protocol refinement

Monolithic protocol refinement = *given* a CSM, *sub*protocol fidelity + deadlock freedom

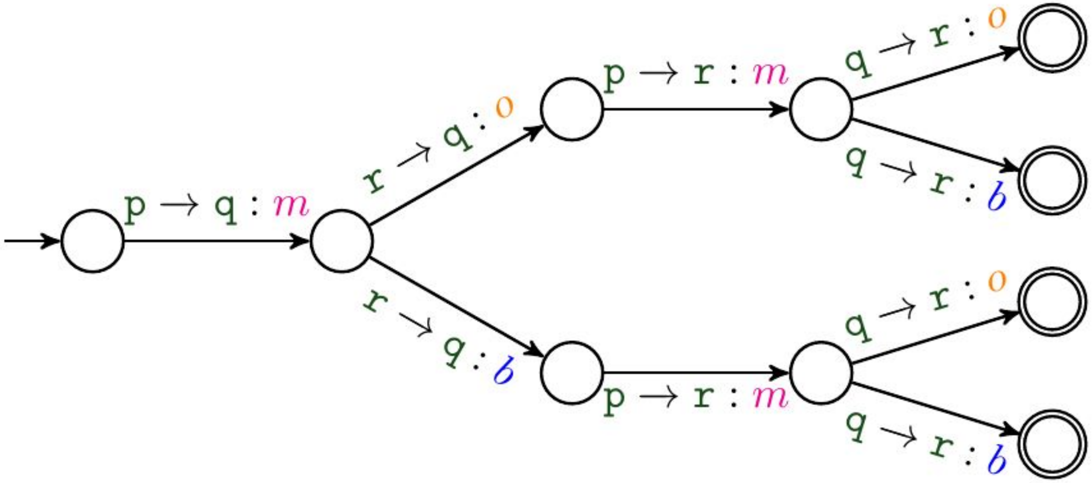


protocol refines?

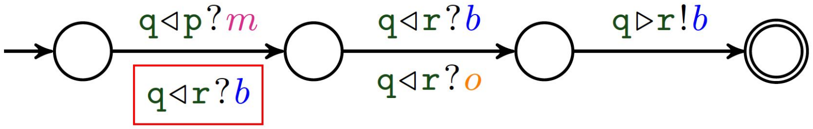


1) CSM language $\subseteq$ global type language
2) CSM is deadlock-free

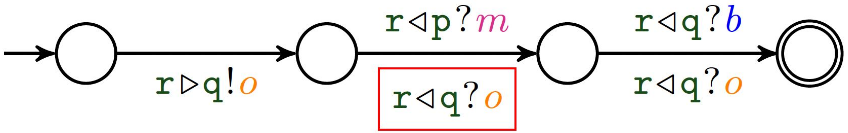
MST monolithic protocol refinement



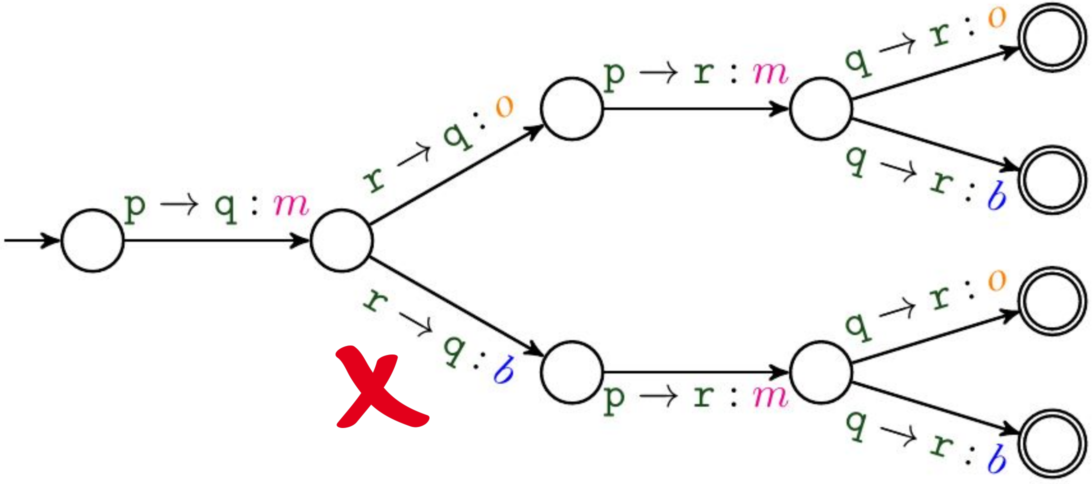
A_q



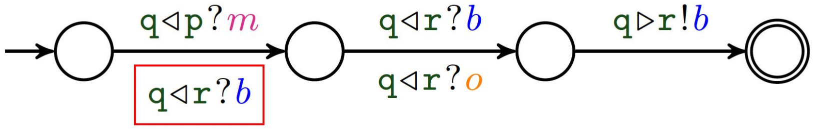
A_r



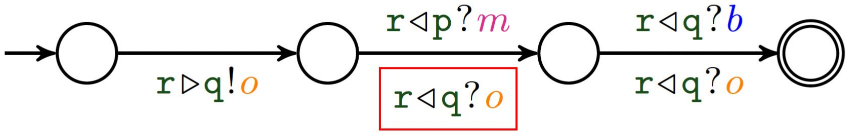
MST monolithic protocol refinement



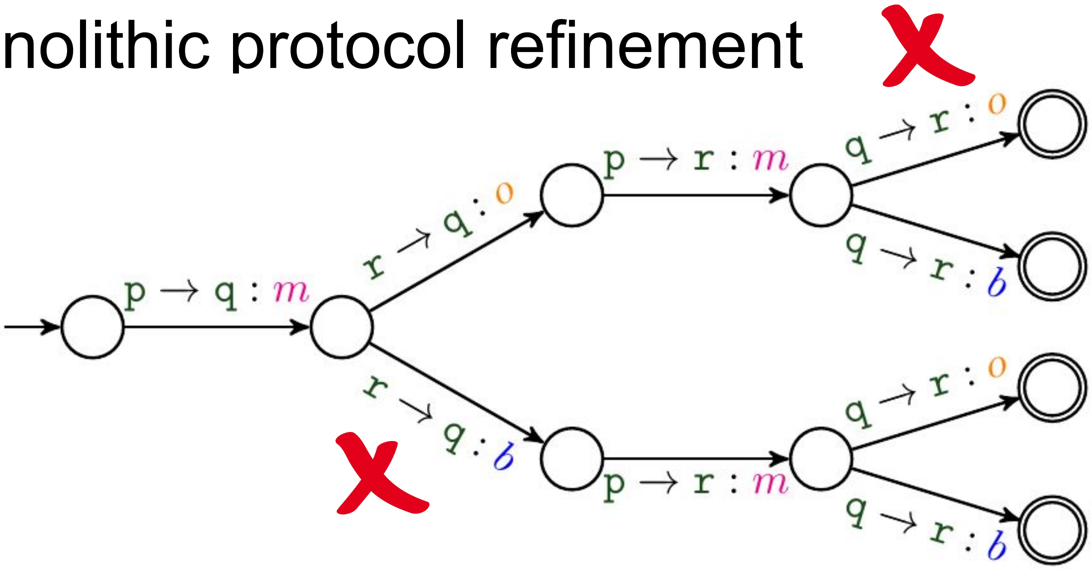
A$_q$



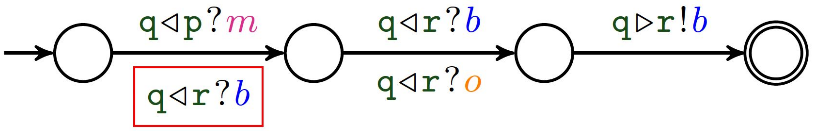
A$_r$



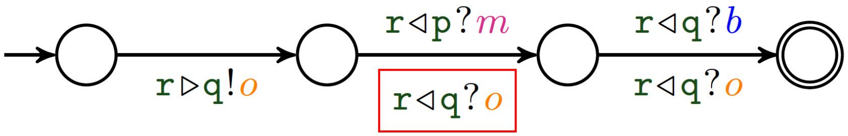
MST monolithic protocol refinement



A_q



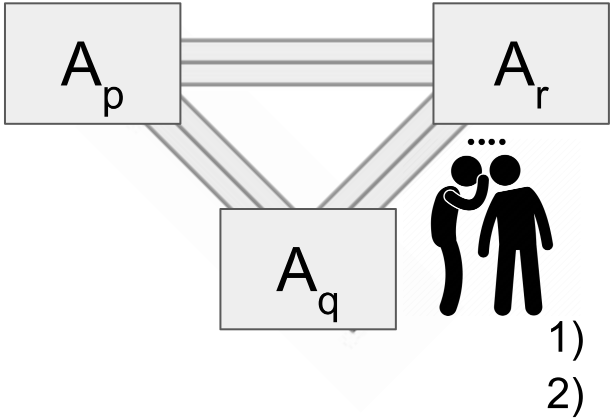
A_r



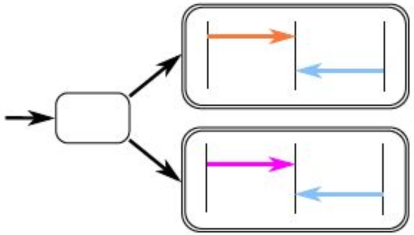
MST monolithic protocol refinement

PSPACE-hard, non-compositional

Monolithic protocol refinement = *given* a CSM, *sub*protocol fidelity + deadlock freedom



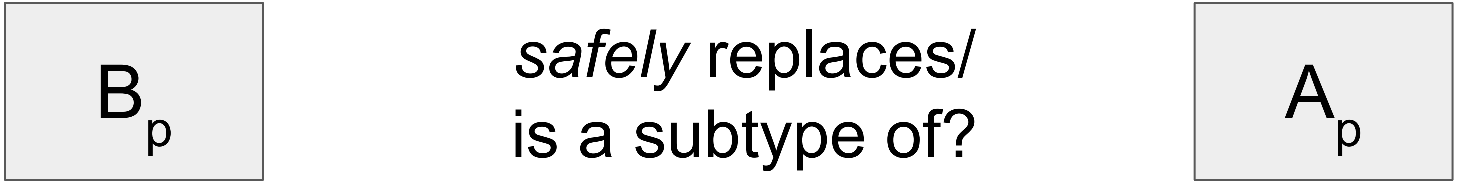
protocol refines?



1) CSM language ⊆ global type language
2) CSM is deadlock-free

MST subtyping

Subtyping = can B_p *safely* replace A_p?



B_p

safely replaces/
is a subtype of?

A_p

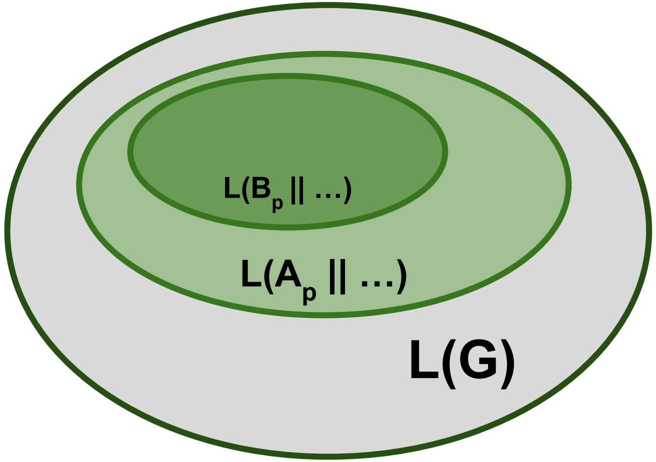
MST subtyping: existing work

Subtyping = can B_p *safely* replace A_p?

Our work provides:

- A stronger notion of safety:

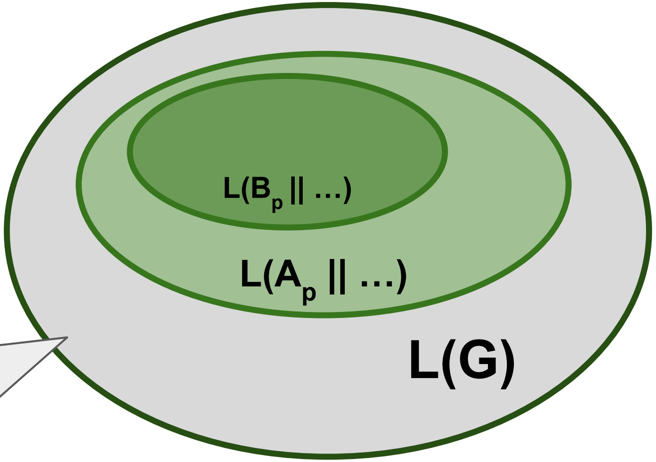
  language inclusion + deadlock freedom



MST subtyping: existing work

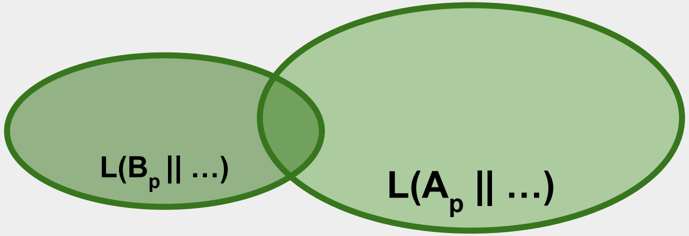
Subtyping = can B_p *safely* replace A_p?

Our work provides:

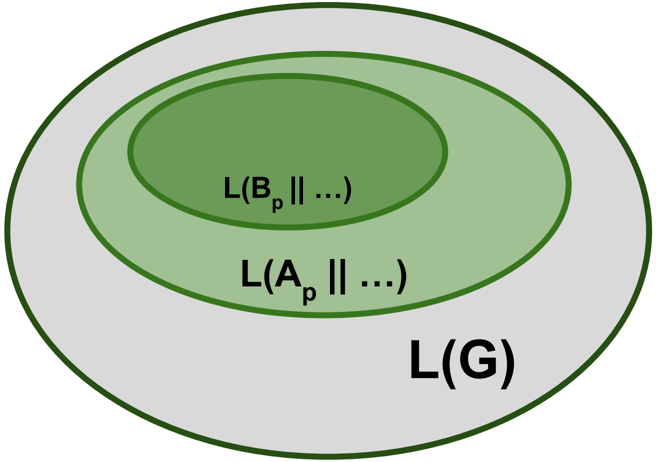


Brief digression on terminology:

"Asynchronous subtyping"



MST subtyping: existing work



Subtyping = can B_p *safely* replace A_p?

Our work provides:

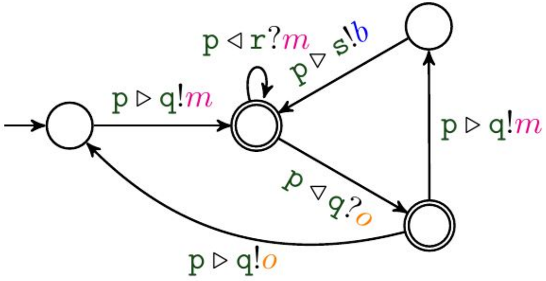
- A stronger notion of safety:

  language inclusion + deadlock freedom

- A more expressive implementation model

  than [Lange and Yoshida, TACAS'16] [Ghilezan et al., POPL'21]

  - Mixed choice
  - Final states with outgoing transitions
  - Unrestricted control flow



MST subtyping: existing work

Subtyping = can B_p *safely* replace A_p?

Our work provides:

- A stronger notion of safety:

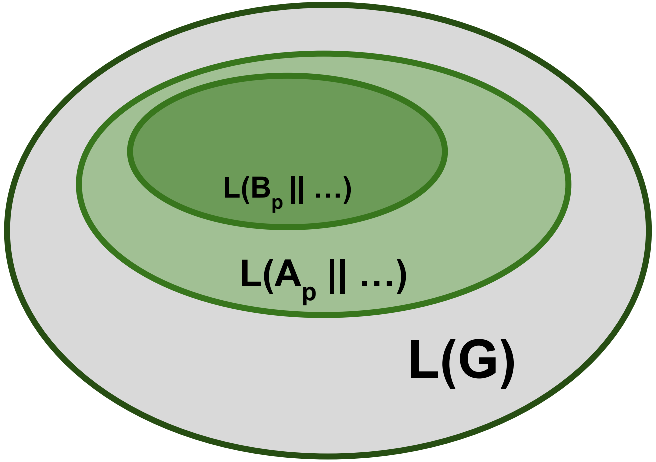
  language inclusion + deadlock freedom

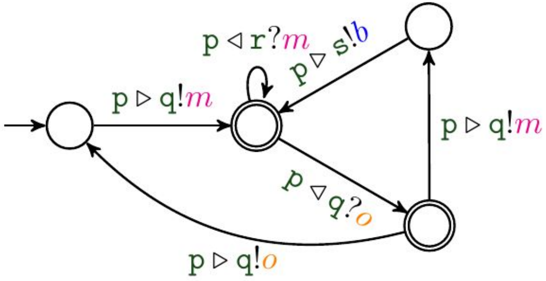
- A more expressive implementation model

  than [Lange and Yoshida, TACAS'16] [Ghilezan et al., POPL'21]

  - Mixed choice
  - Final states with outgoing transitions
  - Unrestricted control flow

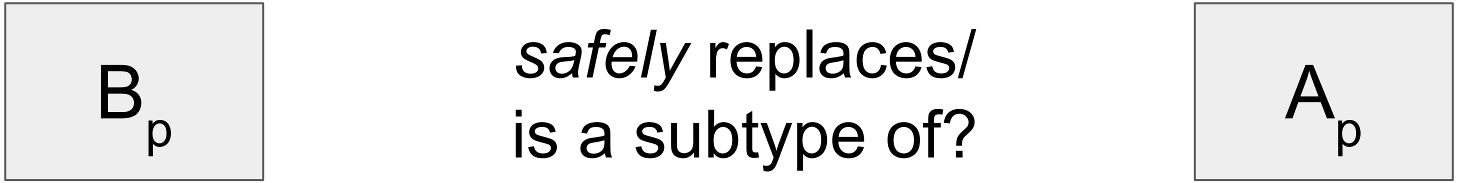
- A **context-dependent** subtyping relation





MST subtyping

Subtyping = can B_p *safely* replace A_p?



B_p

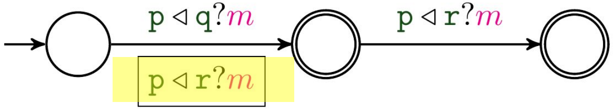
safely replaces/
is a subtype of?

A_p

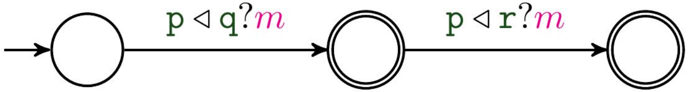
Folkloric subtyping: *"add receives, remove sends"*

MST subtyping

Subtyping = can B_p *safely* replace A_p?

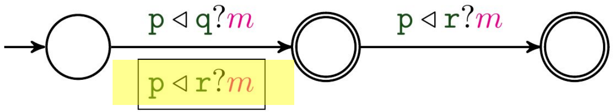


safely replaces/
is a subtype of?

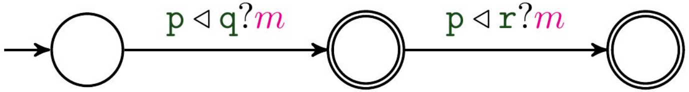


MST subtyping

Subtyping = can B_p *safely* replace A_p?



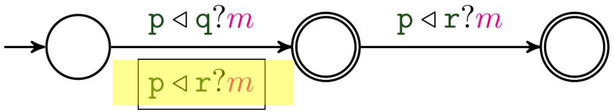
safely replaces/
is a subtype of?



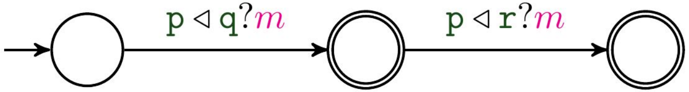
It depends!

MST subtyping

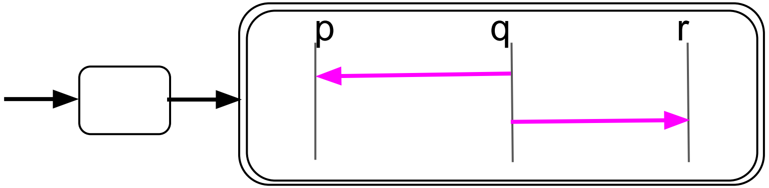
Subtyping = can B_p *safely* replace A_p?



safely replaces/
is a subtype of?

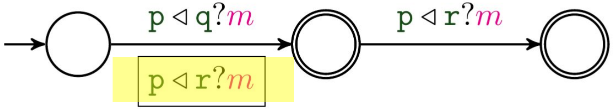


It depends! ✓

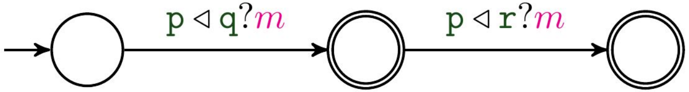


MST subtyping

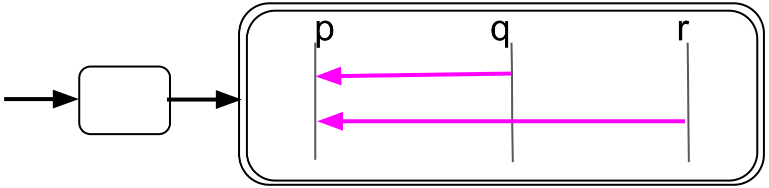
Subtyping = can B_p *safely* replace A_p?



safely replaces/
is a subtype of?



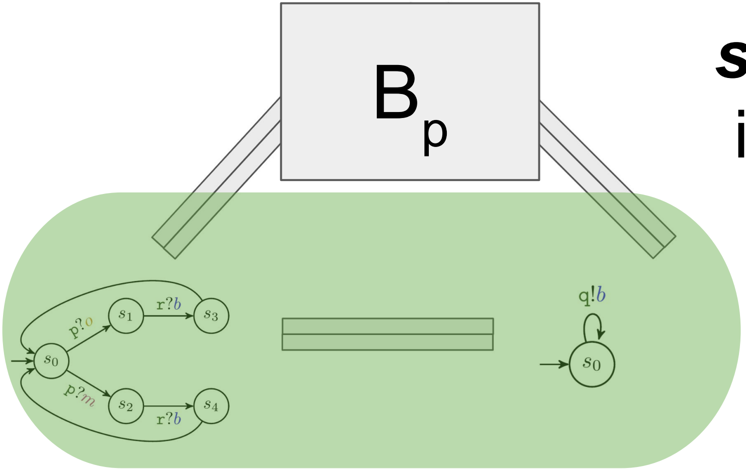
It depends! ✗



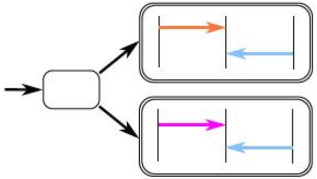
MST protocol refinement (subtyping)

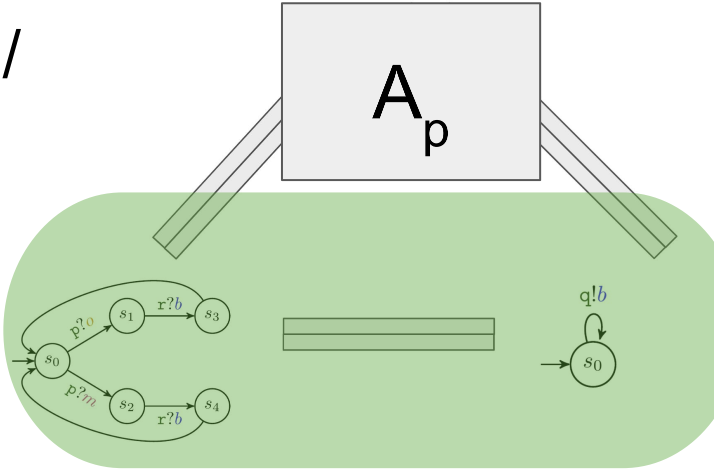
polynomial, compositional, context-dependent

Protocol refinement = **for all well-behaved contexts under G**, can B_p safely replace A_p?



safely$_G$ replaces/
is a subtype of?





A_p
s_0
s_1
s_2
s_3
s_4
p?o
p?m
r?b
r?b
q!b
s_0