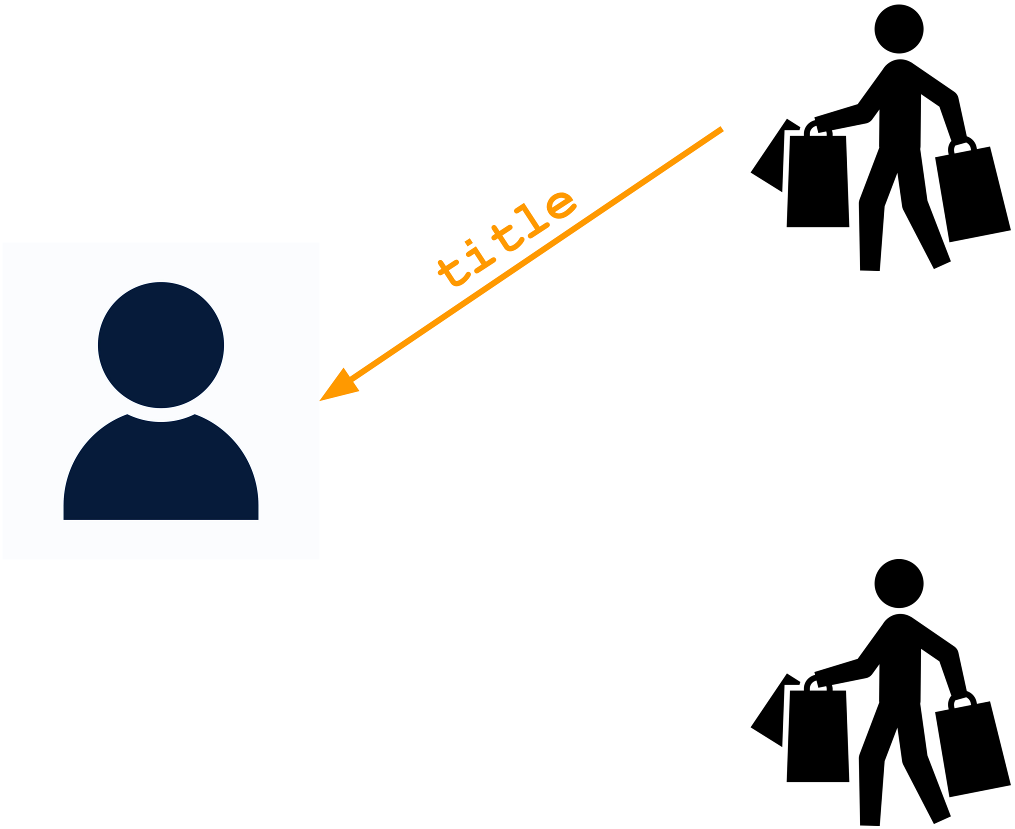
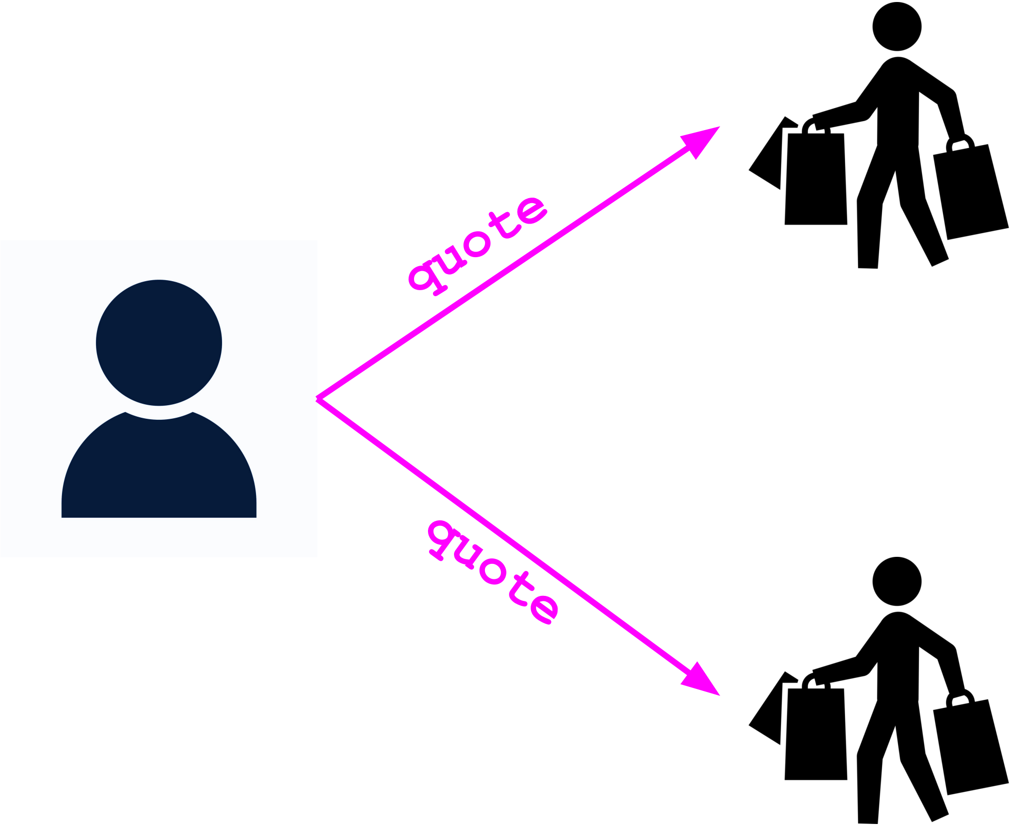
Complete
Multiparty Session Type Projection
with Automata

Elaine Li Felix Stutz Thomas Wies Damien Zufferey

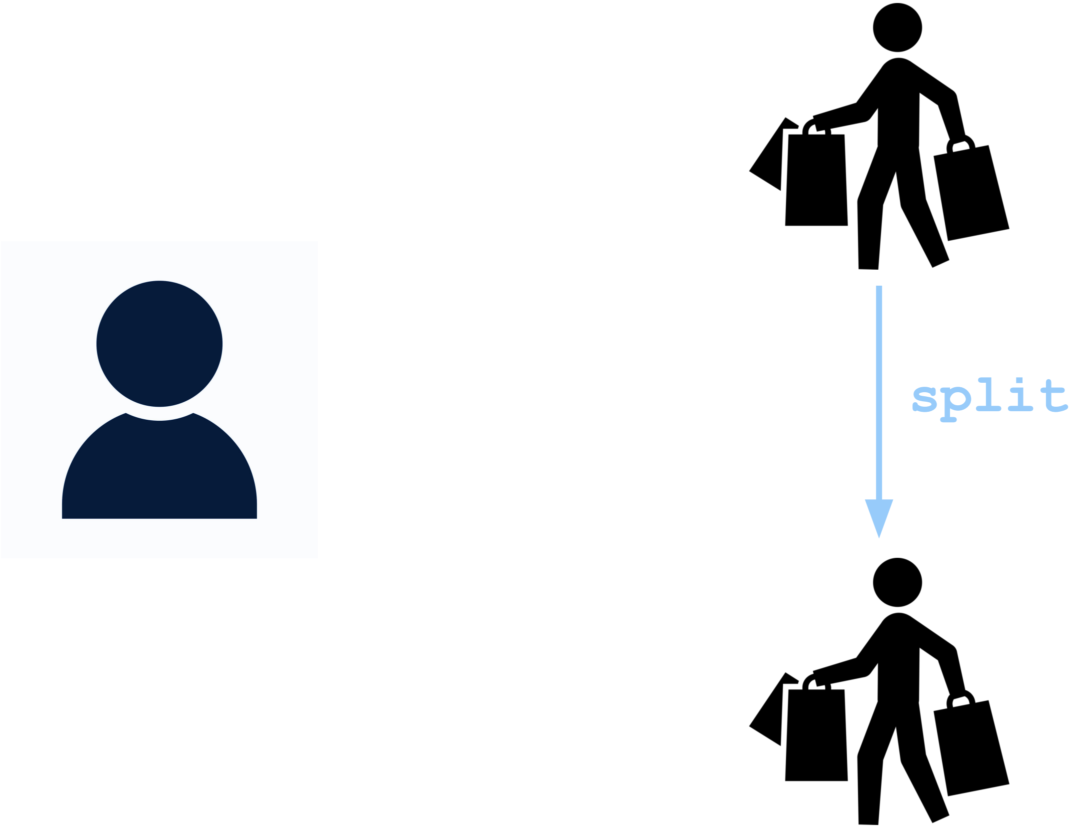
Multiparty session types: two-buyer protocol



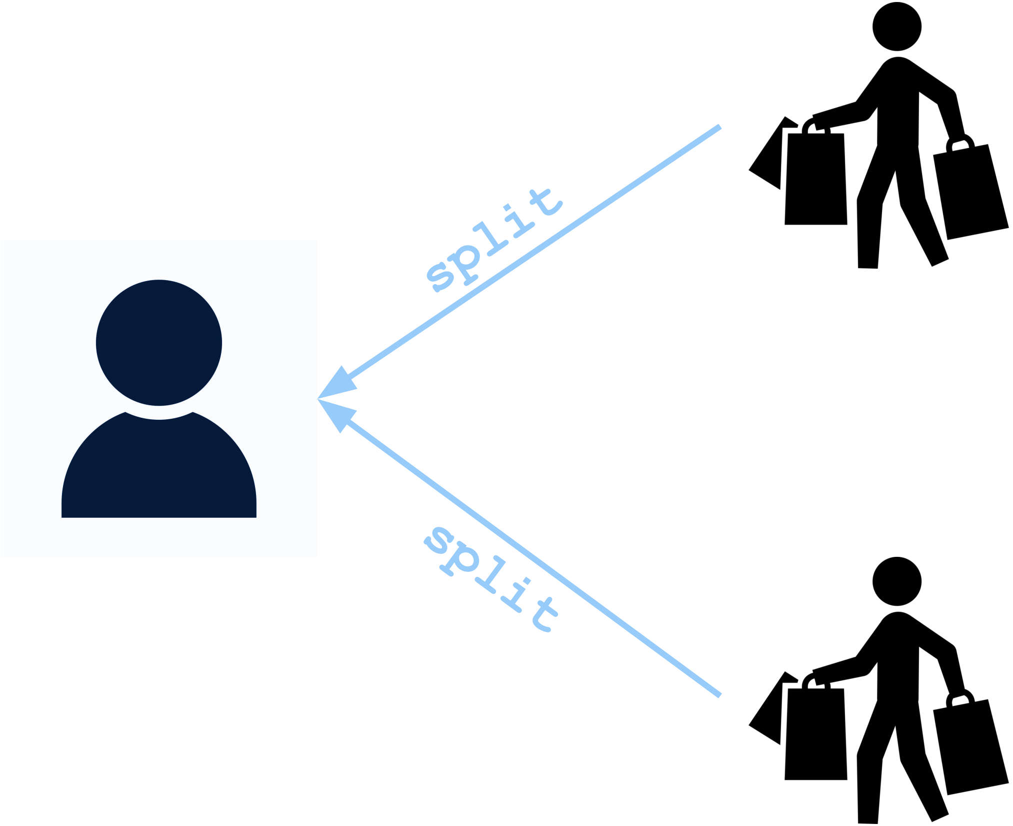
Multiparty session types: two-buyer protocol



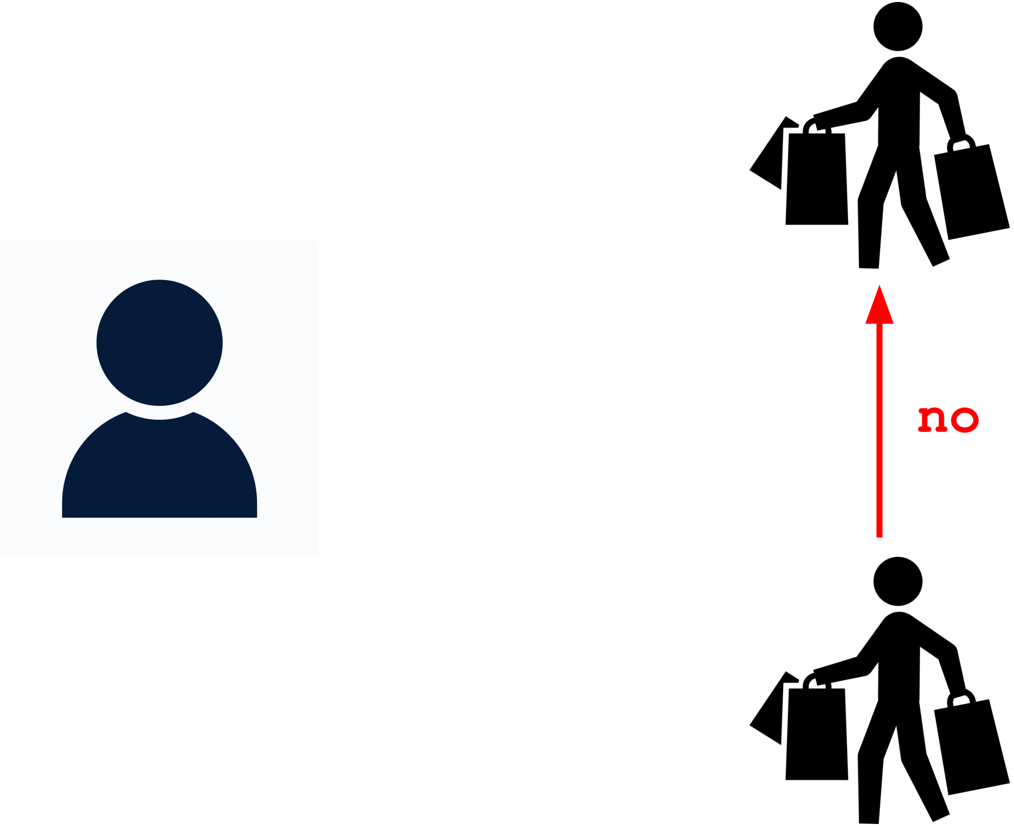
Multiparty session types: two-buyer protocol



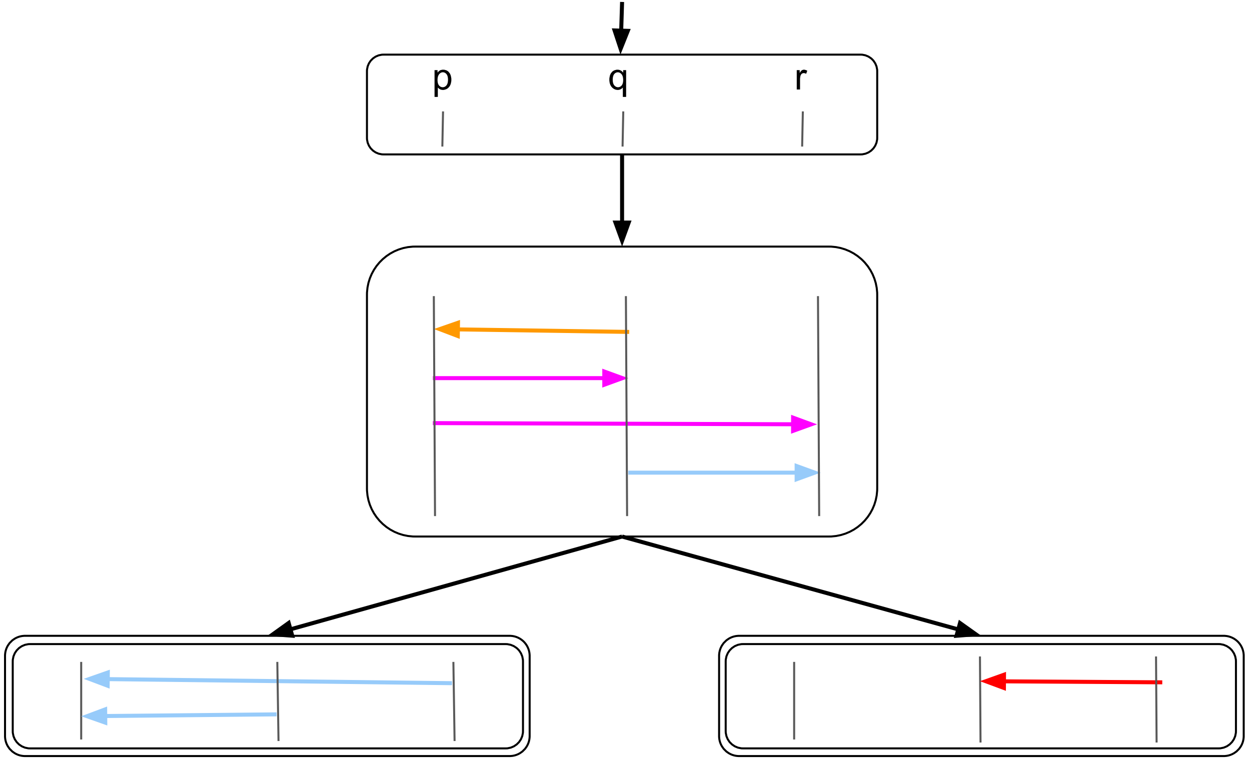
Multiparty session types: two-buyer protocol



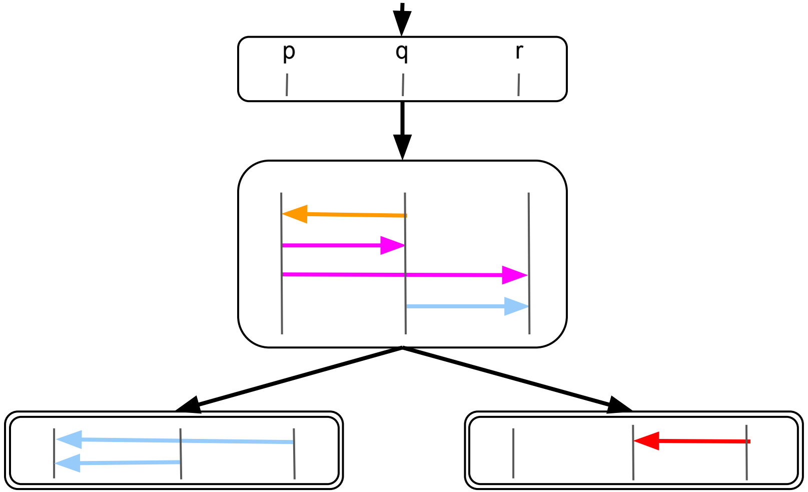
Multiparty session types: two-buyer protocol



Multiparty session types

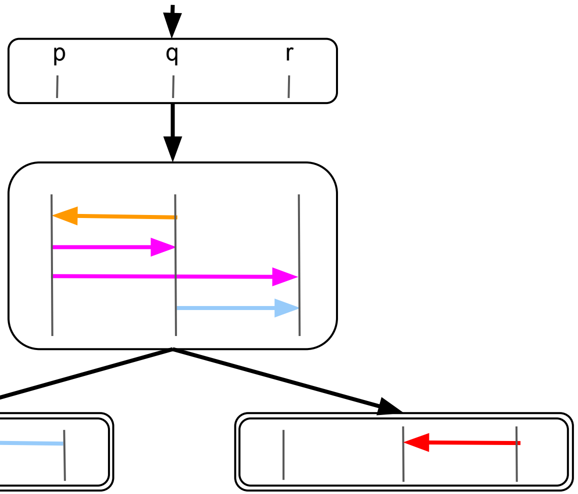


MST semantics



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

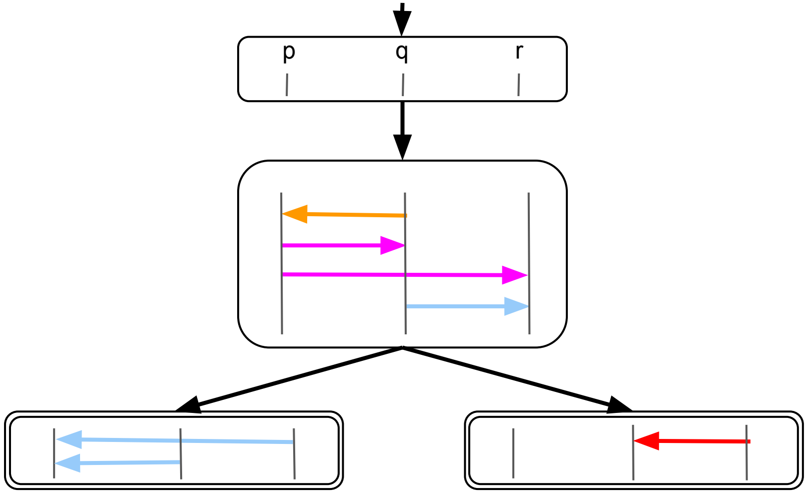
MST semantics



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

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

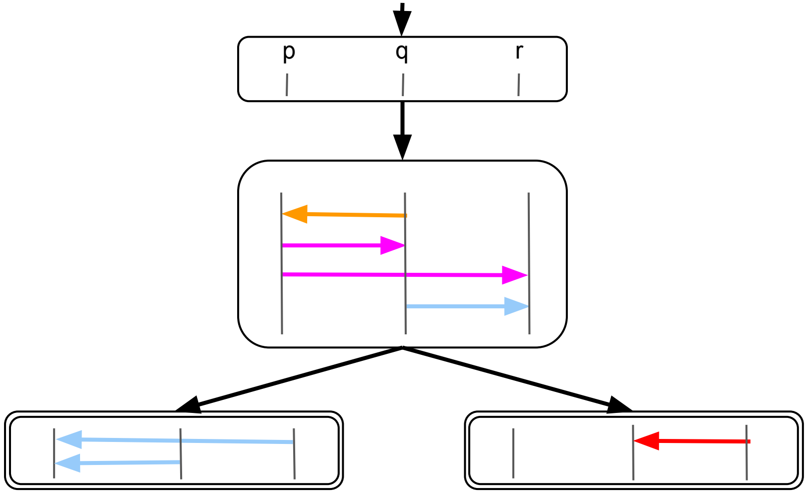
MST semantics



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

MST semantics



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

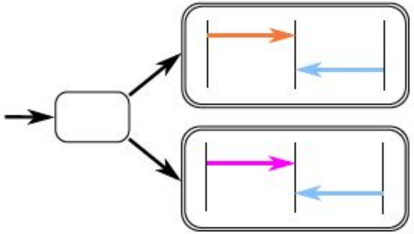
Asynchronous $\mathtt{q} \triangleright \mathtt{p}!o \cdot \mathtt{p} \triangleleft \mathtt{q}?o \cdot \mathtt{p} \triangleright \mathtt{q}!m \cdot \mathtt{q} \triangleleft \mathtt{p}?m \cdot \mathtt{p} \triangleright \mathtt{r}!m \cdot \mathtt{r} \triangleleft \mathtt{p}?m \cdot \mathtt{q} \triangleright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

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 implementability

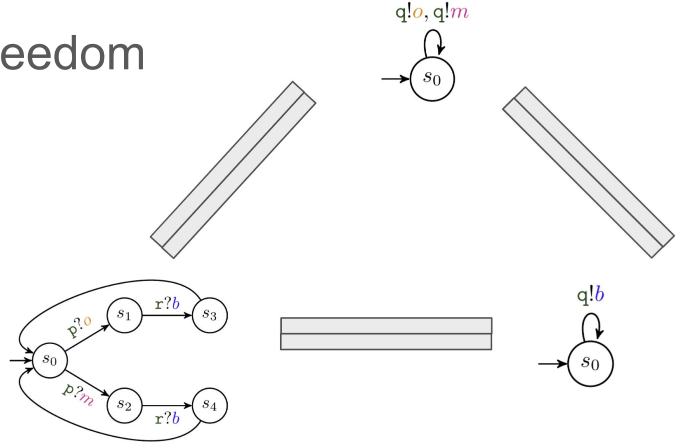
Implementation model: communicating state machines (CSMs)

Implementability = protocol fidelity + deadlock freedom



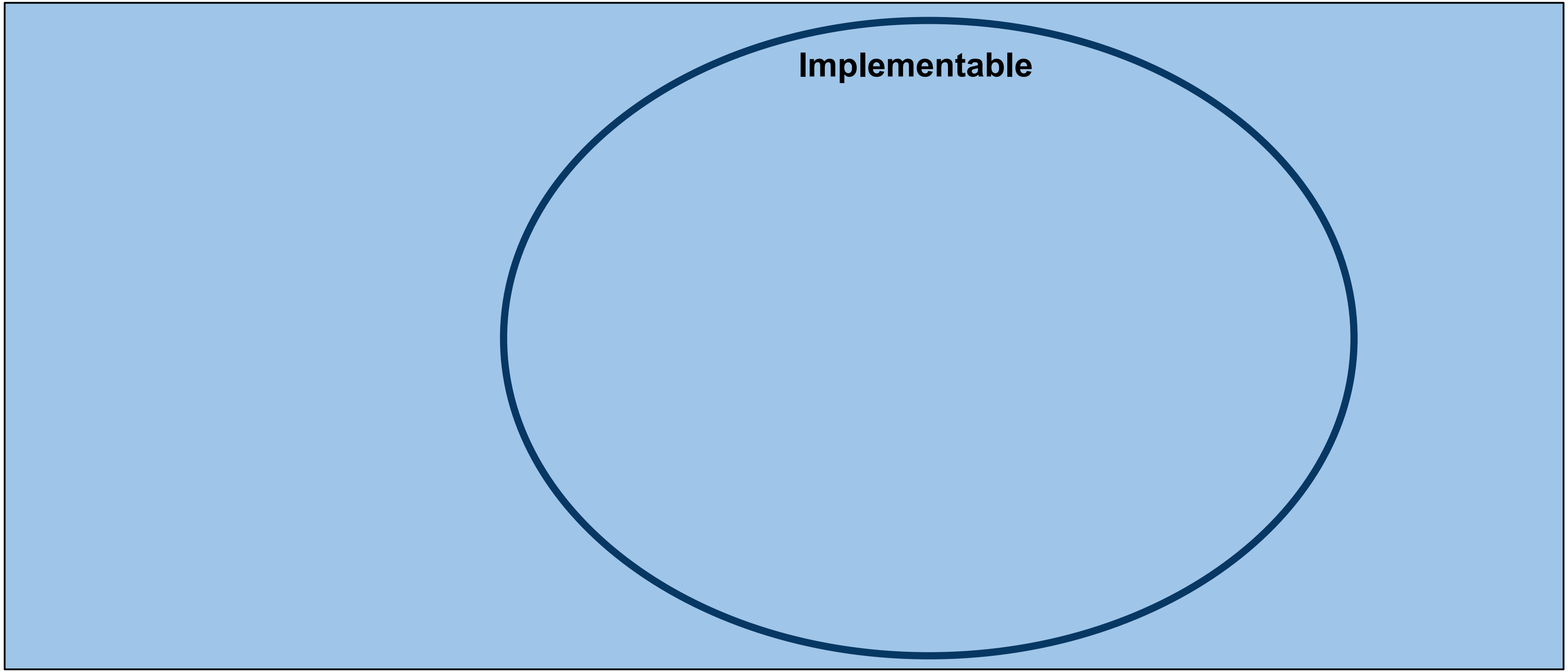
?

projection operator

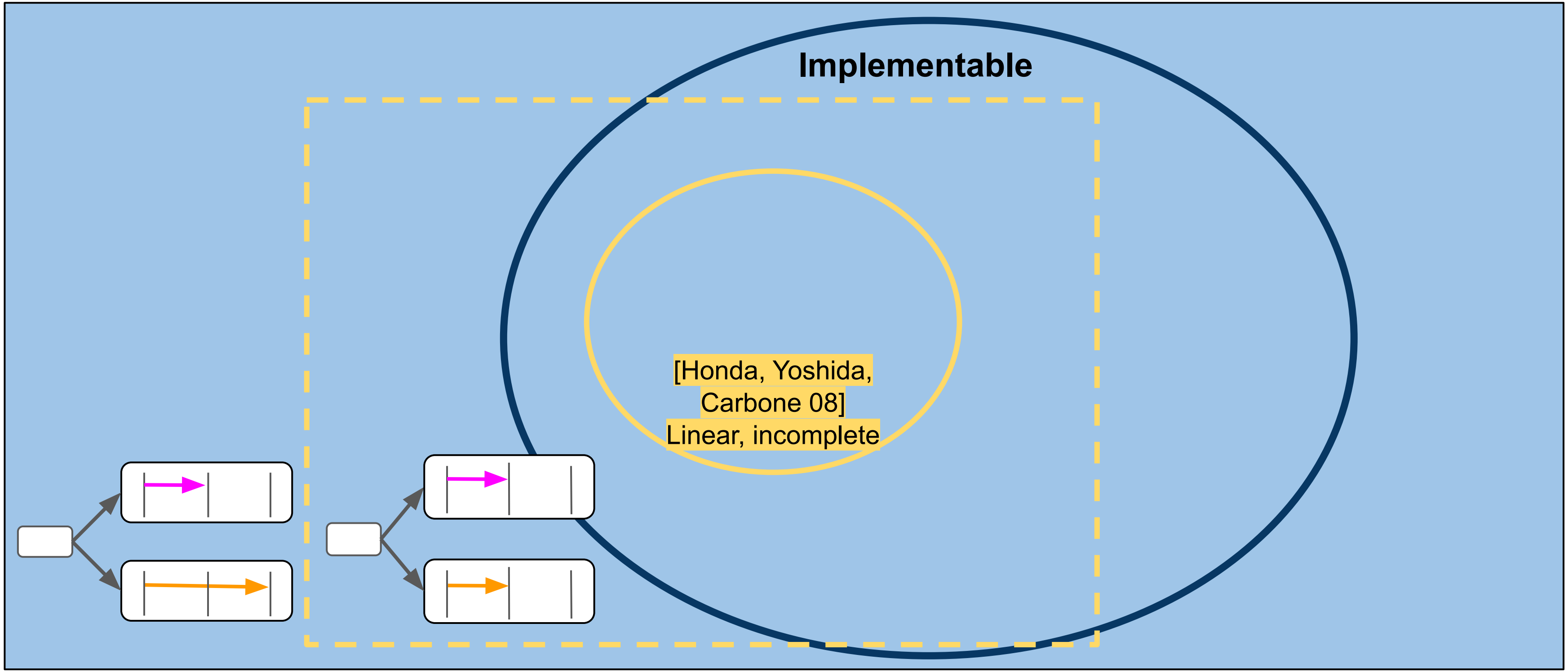


1) CSM language = global type language
2) CSM is deadlock-free

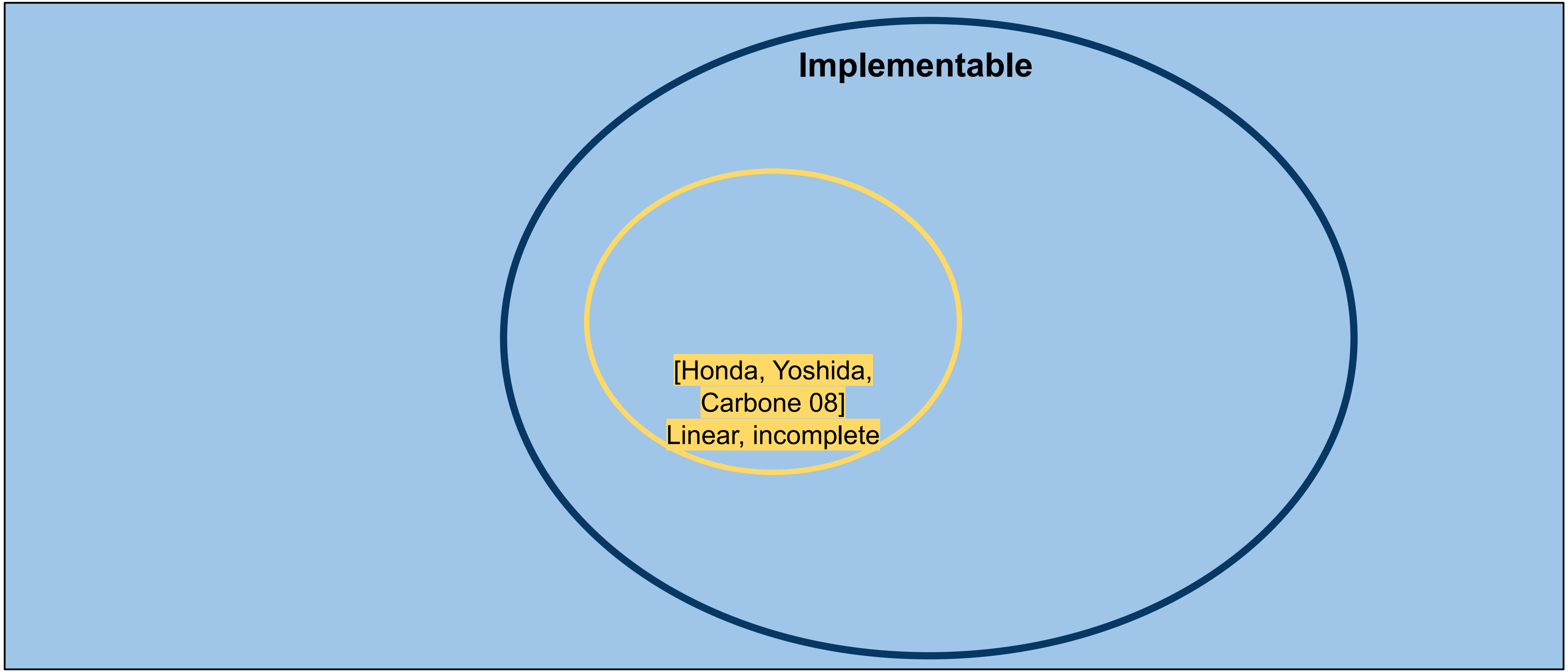
Contextualizing our contribution



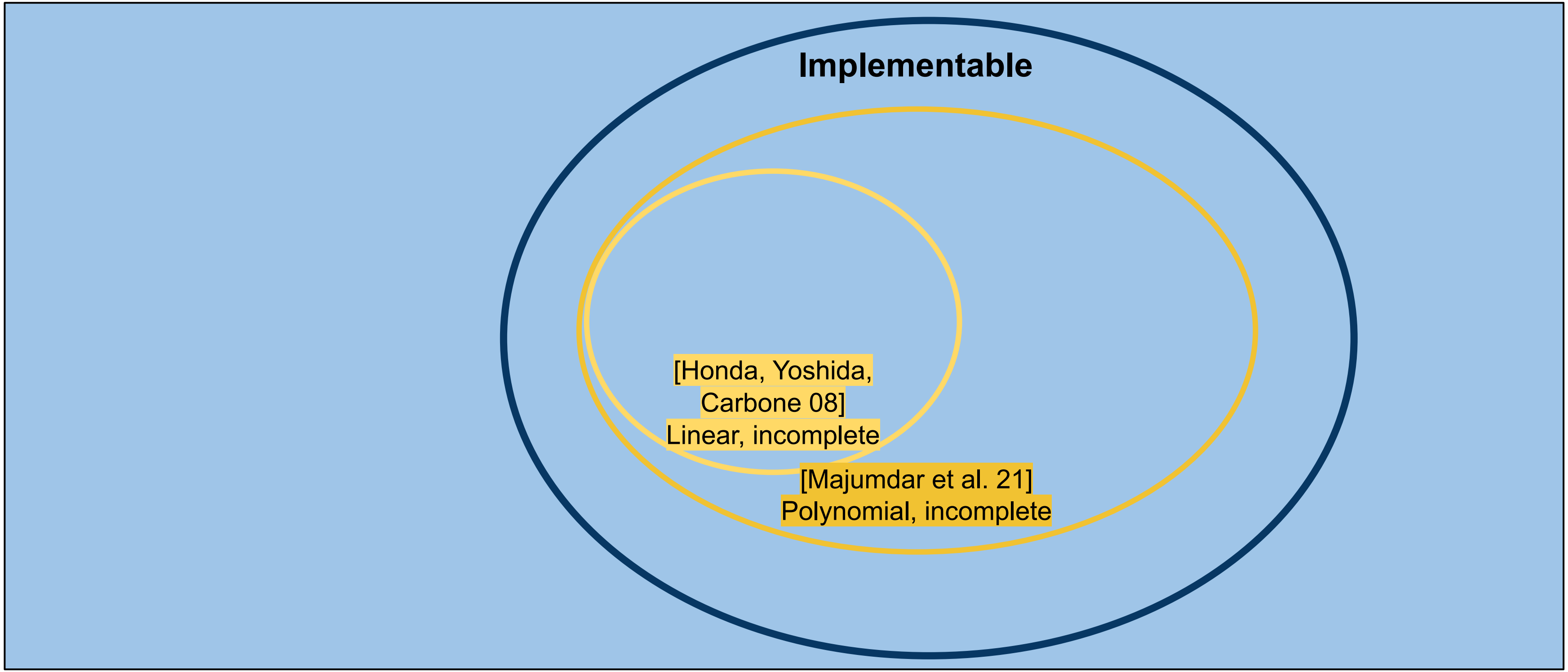
Contextualizing our contribution



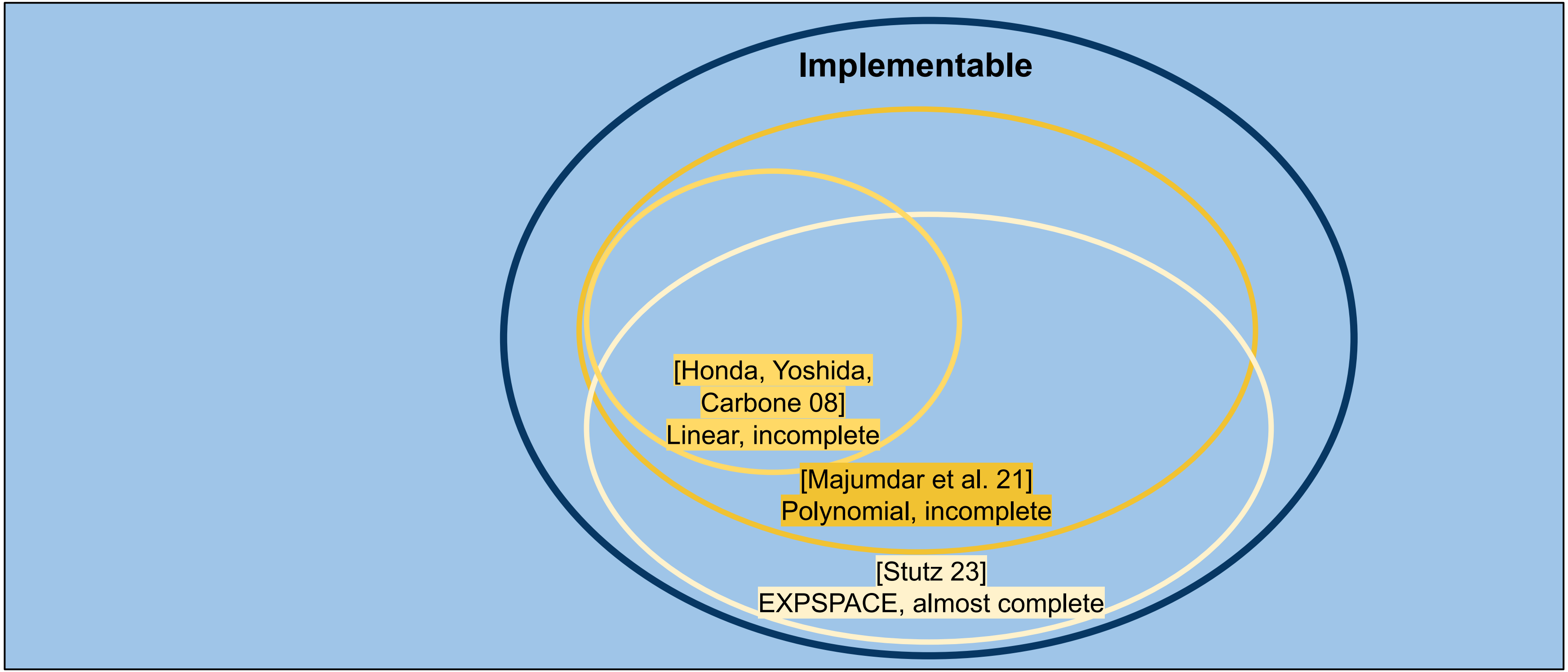
Contextualizing our contribution



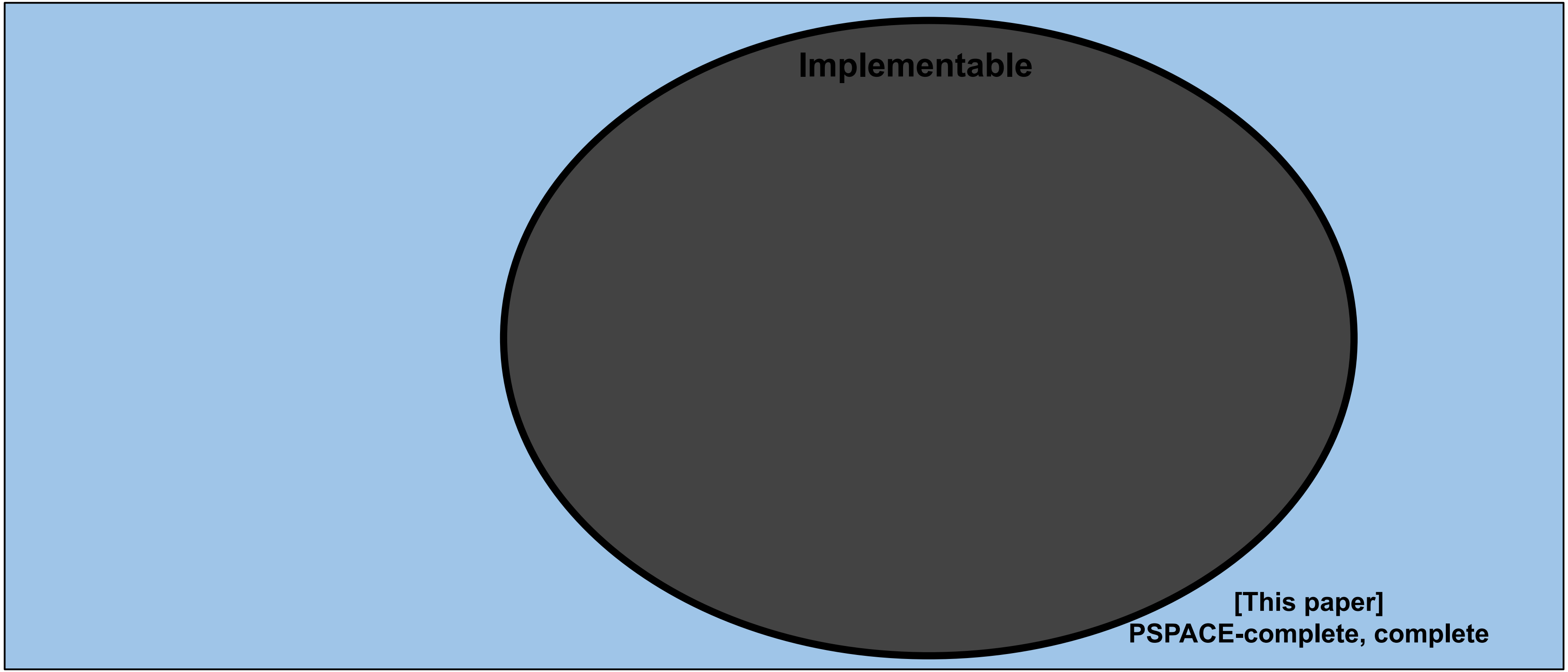
Contextualizing our contribution



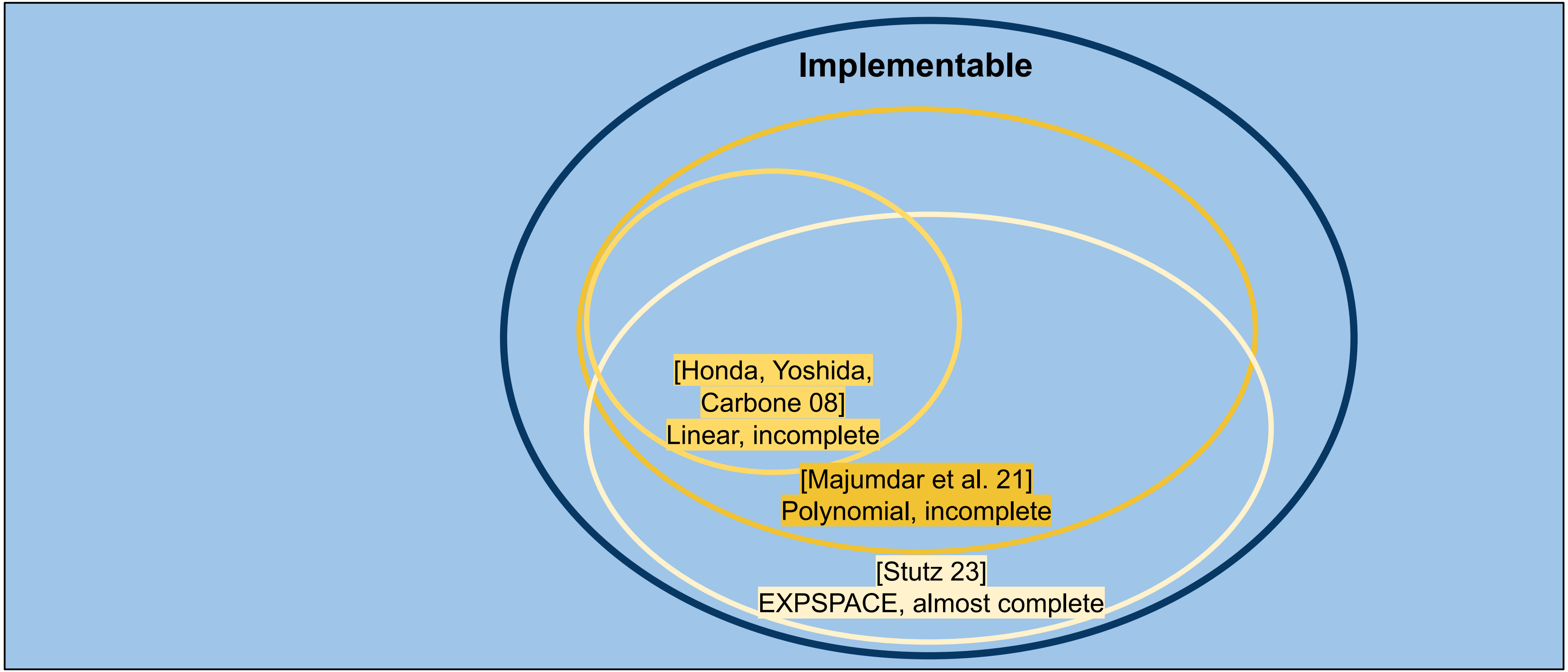
Contextualizing our contribution



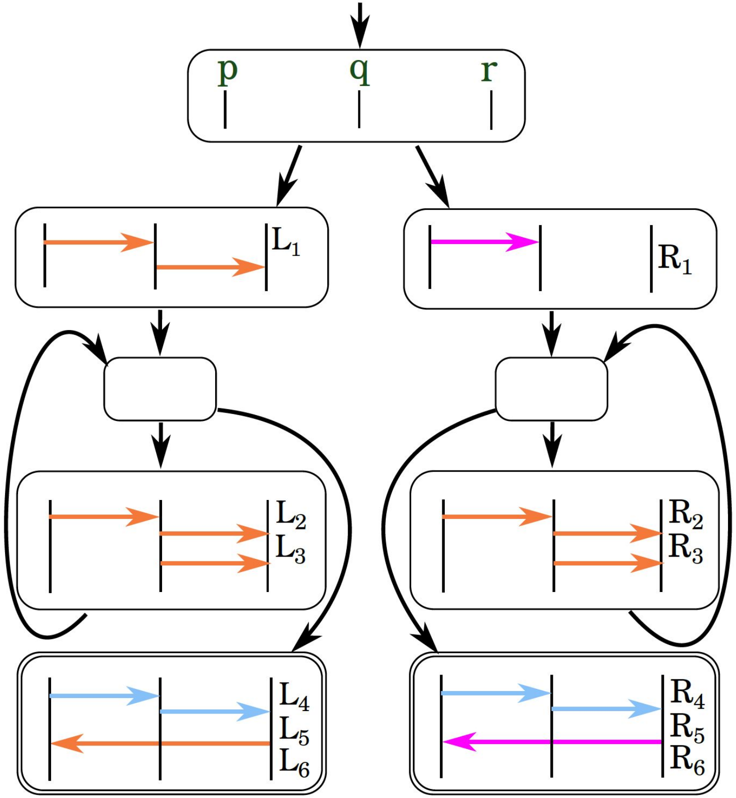
Contextualizing our contribution



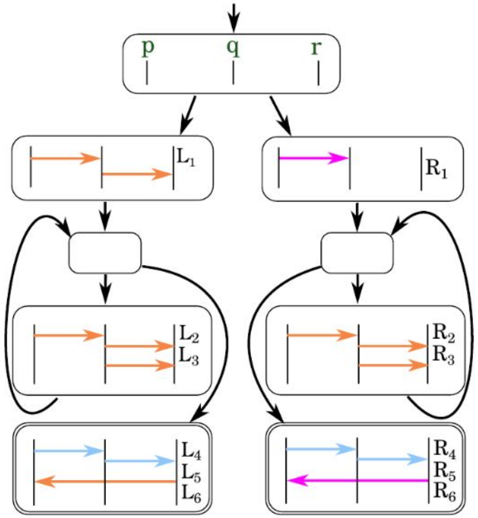
Explaining the completeness gap



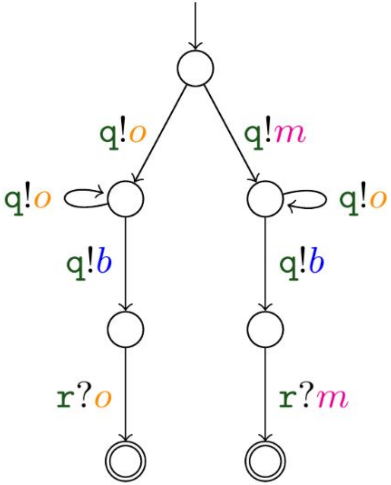
Explaining the completeness gap: odd-even example



Odd-even example

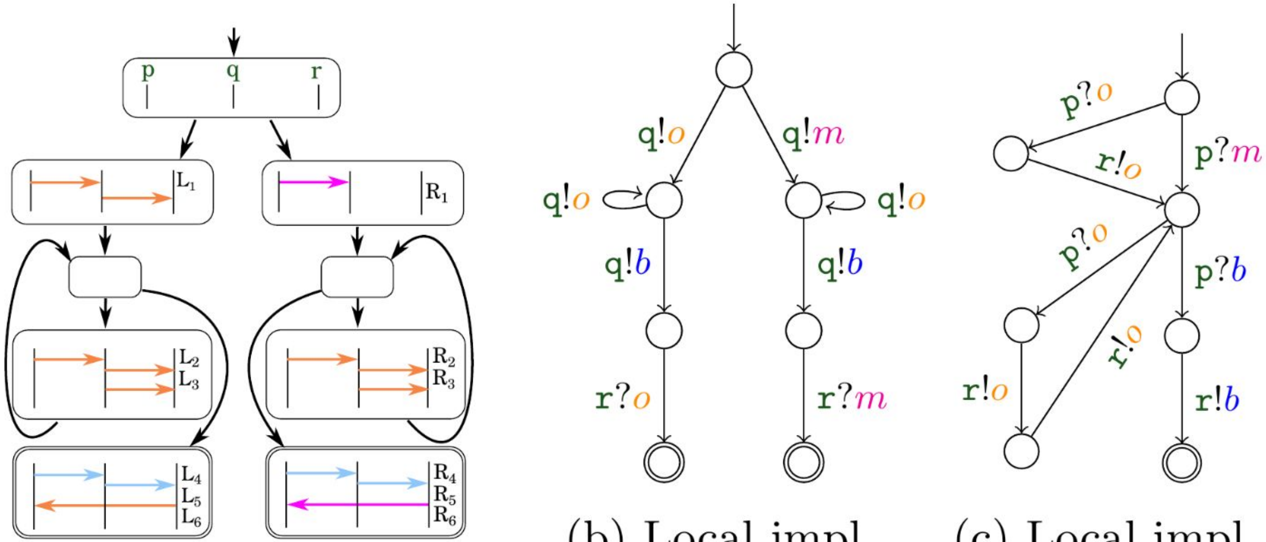


(a) Odd-even protocol



(b) Local impl. for role p

Odd-even example

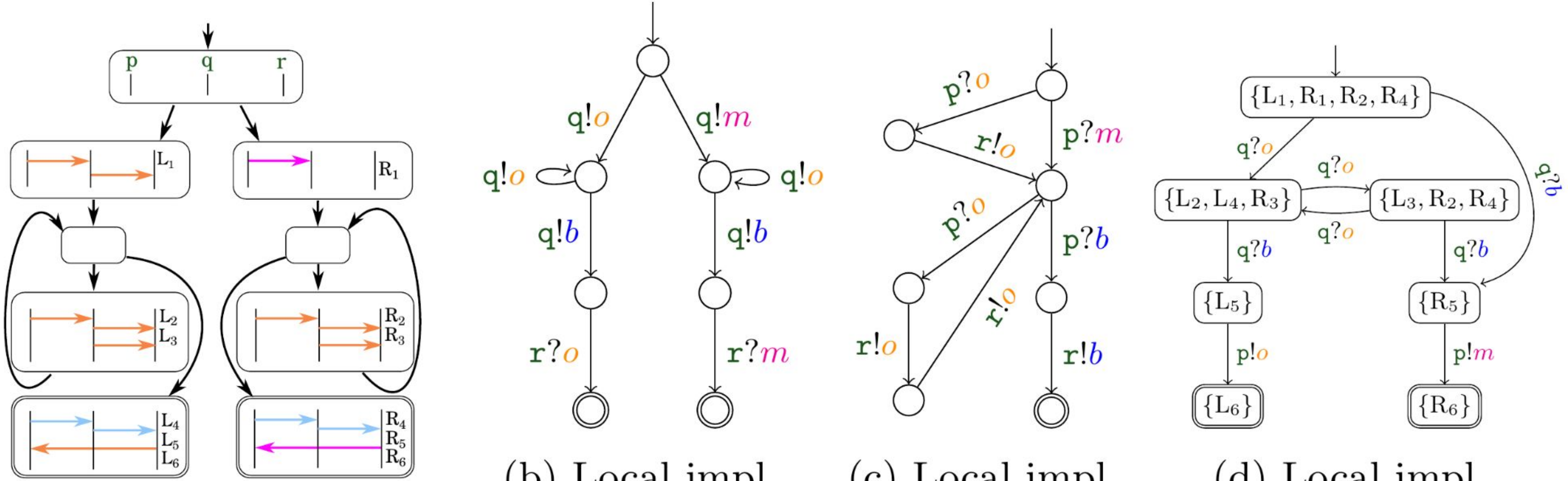


(a) Odd-even protocol

(b) Local impl. for role p

(c) Local impl. for role q

Odd-even example



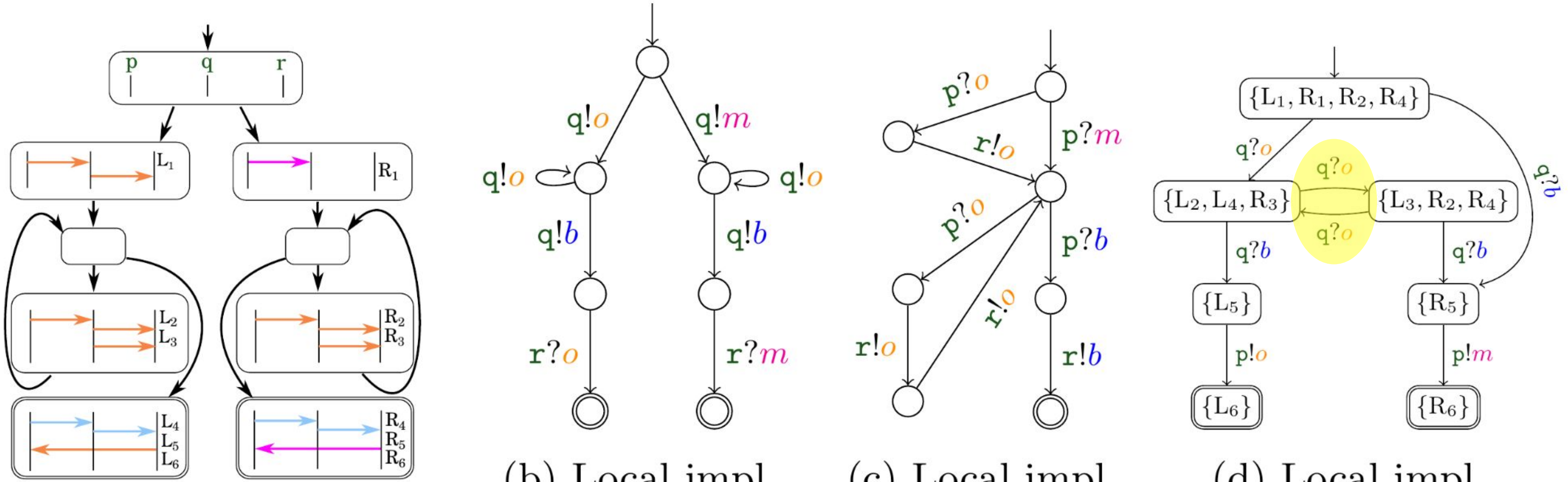
(a) Odd-even protocol

(b) Local impl. for role p

(c) Local impl. for role q

(d) Local impl. for role r

Odd-even example



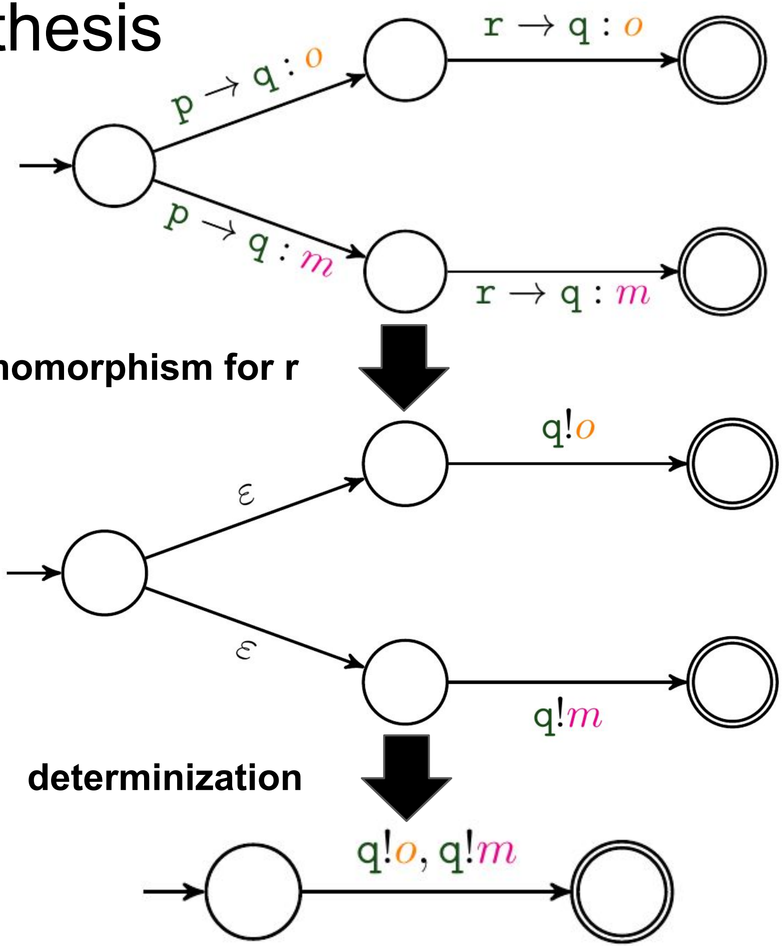
(a) Odd-even protocol

(b) Local impl. for role p

(c) Local impl. for role q

(d) Local impl. for role r

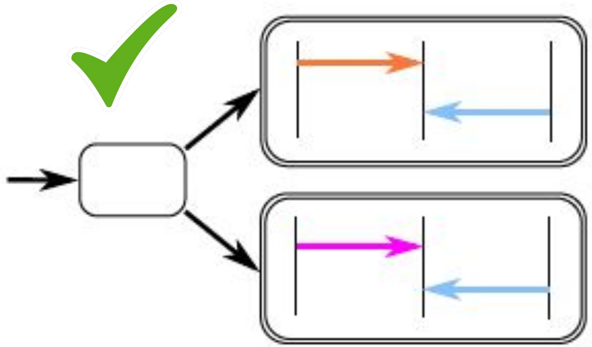
Projection: Synthesis

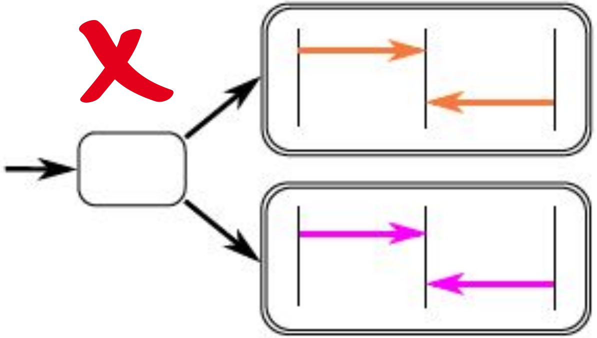


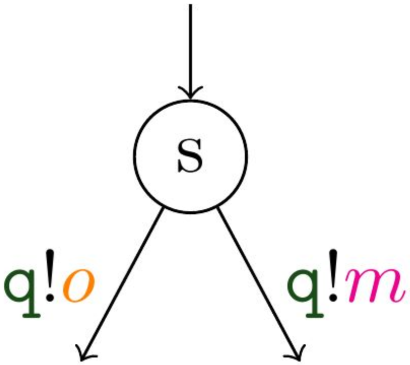
homomorphism for r

determinization

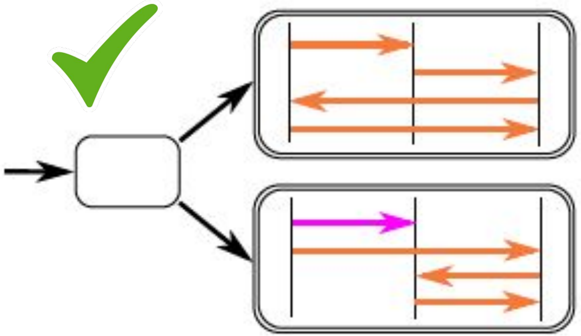
Projection: Checking Implementability – Send

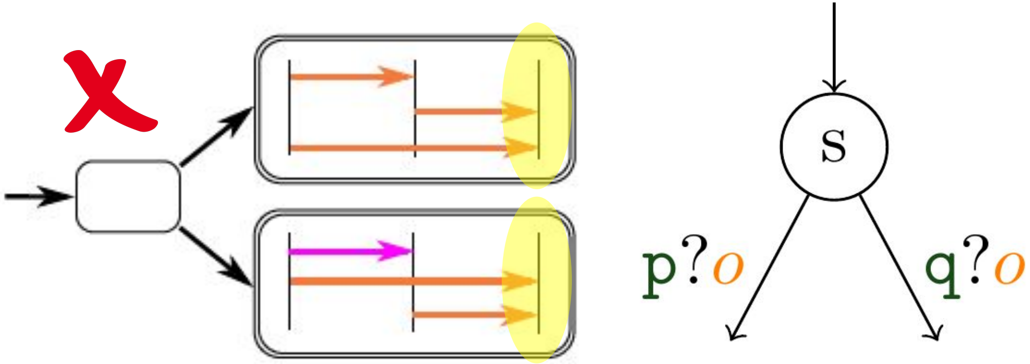






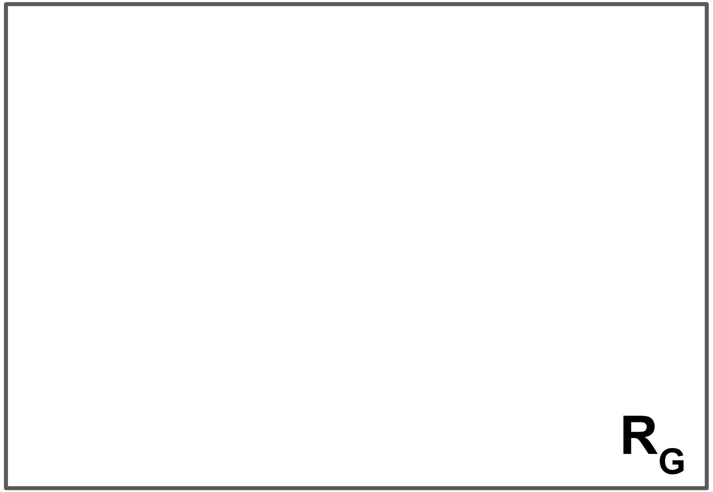
Projection: Checking Implementability – Receive





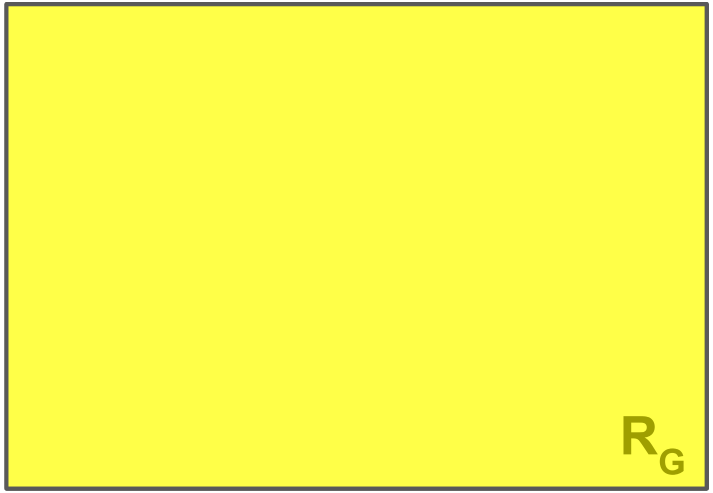
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



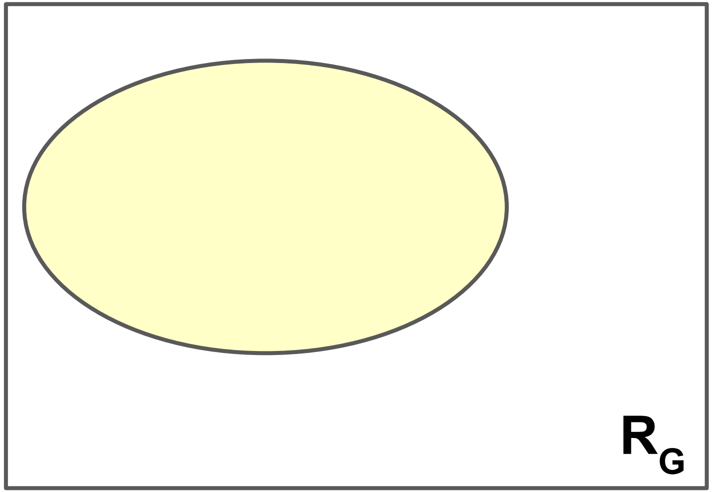
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



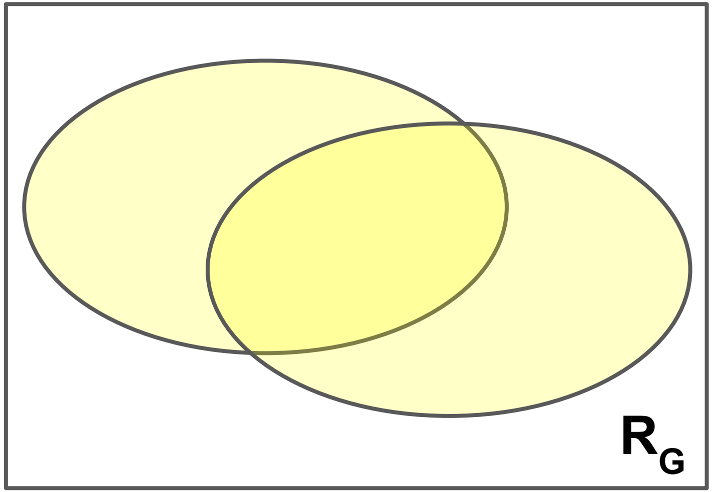
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



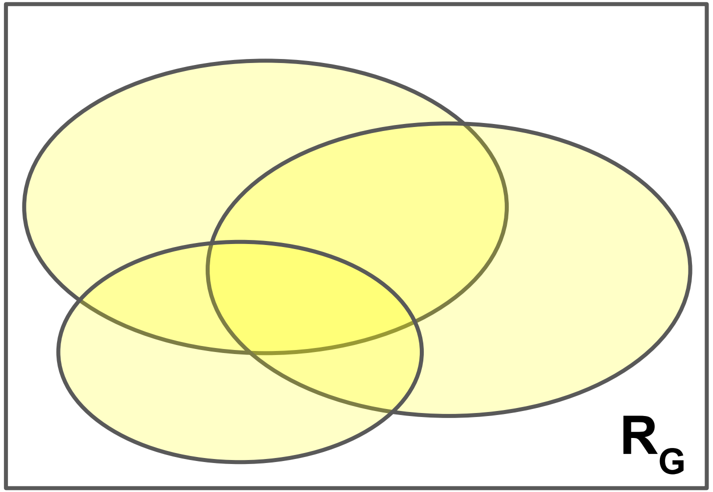
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



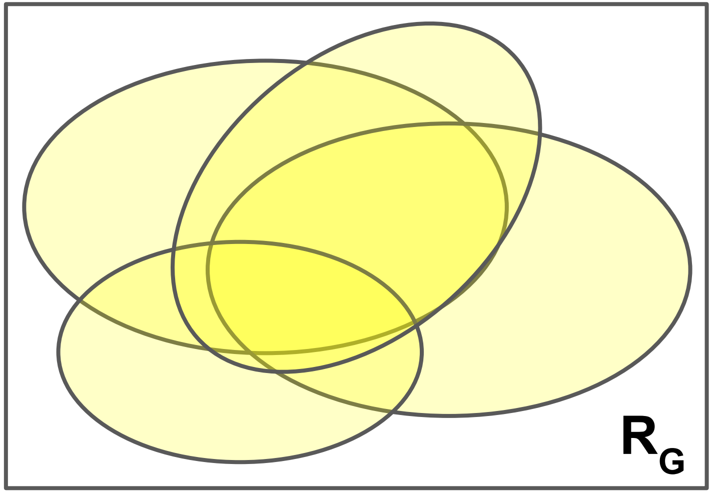
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



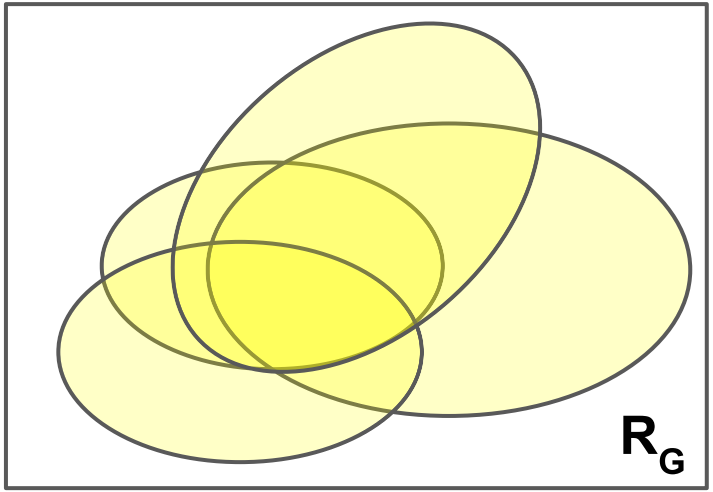
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



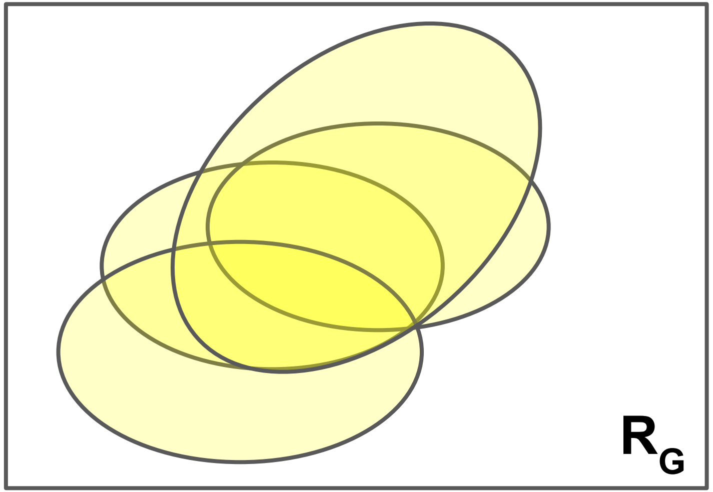
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



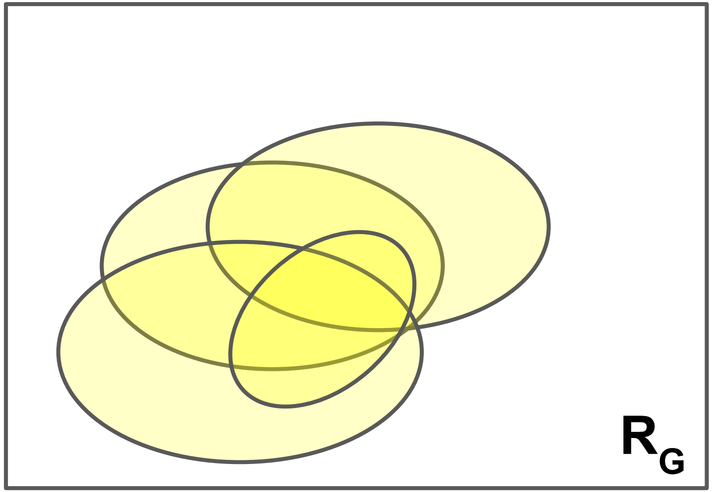
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



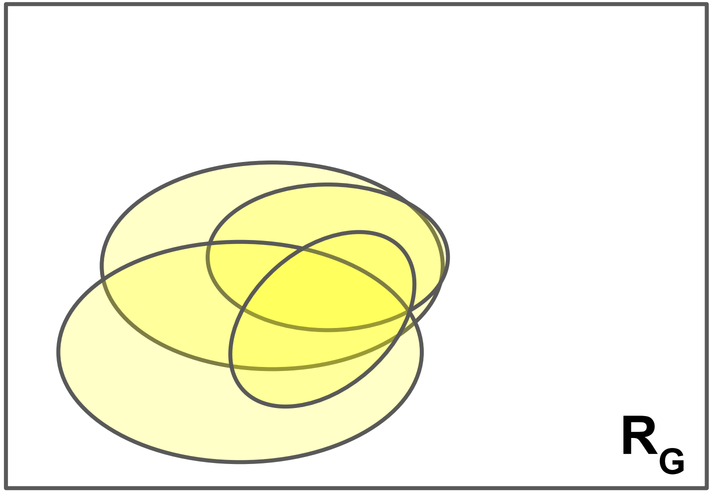
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



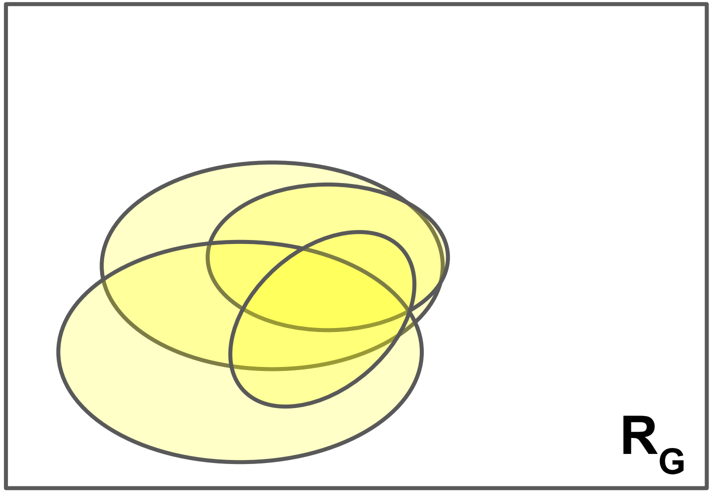
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



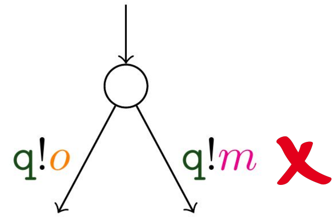
Soundness: Projectable ⇒ Implementable

Inductive invariant: the intersection of "possible runs" for all roles is non-empty



Completeness: Implementable ⇒ Projectable

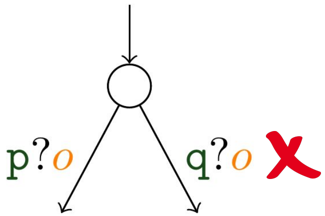
Send Validity



construction of w →

$w \cdot \mathtt{r} \triangleright \mathtt{q}!m$ ✗

Receive Validity

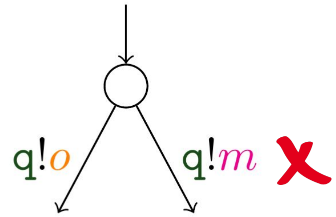


construction of w →

$w \cdot \mathtt{r} \triangleleft \mathtt{q}?o$ ✗

Completeness: Implementable ⇒ Projectable

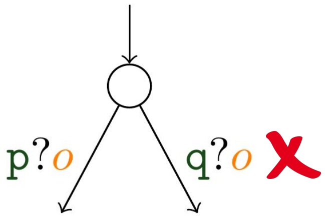
Send Validity



$\xrightarrow{\text{construction of w}}$ $w \cdot \mathrm{r} \triangleright \mathrm{q}!m$ ✗

Corollary. Any implementable global type can be implemented without mixed choice.

Receive Validity



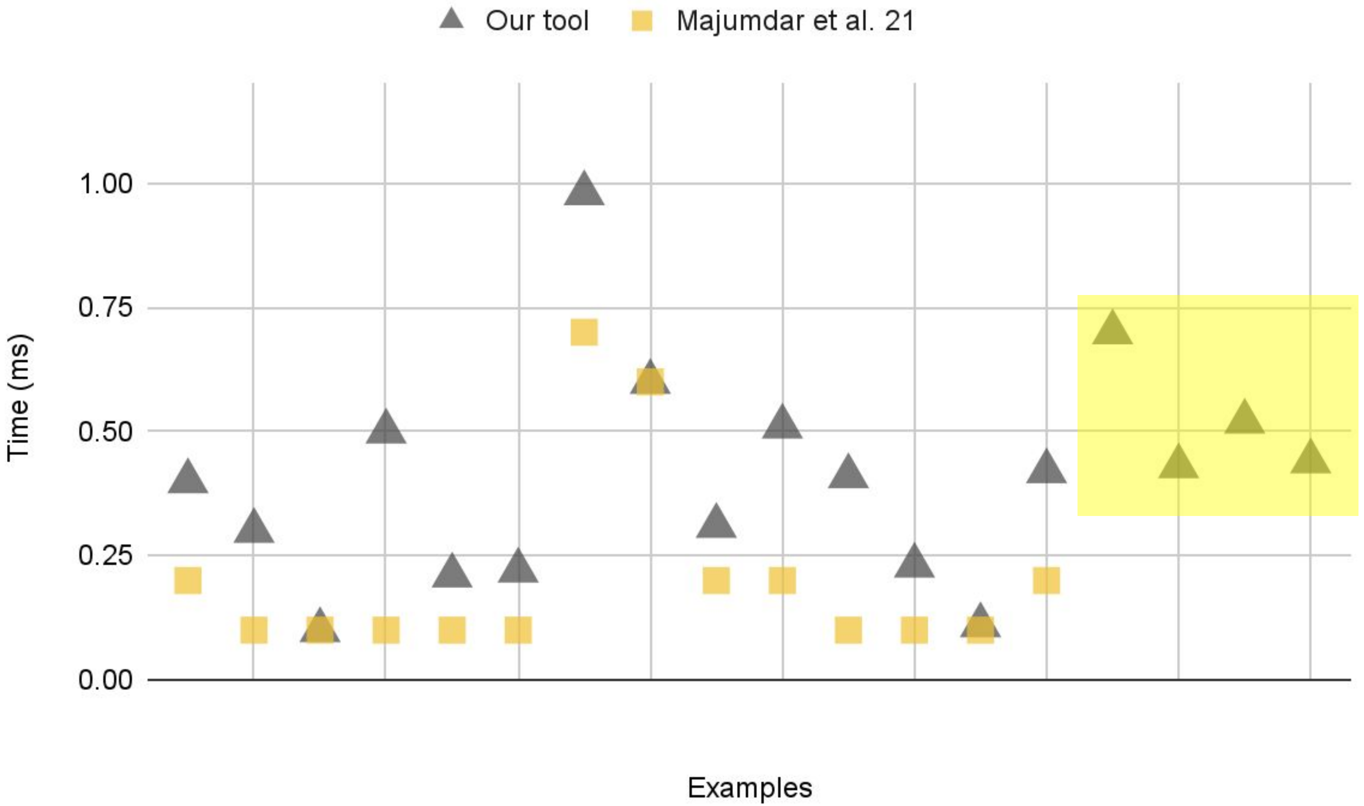
$\xrightarrow{\text{construction of w}}$ $w \cdot \mathrm{r} \triangleleft \mathrm{q}?o$ ✗

Complexity

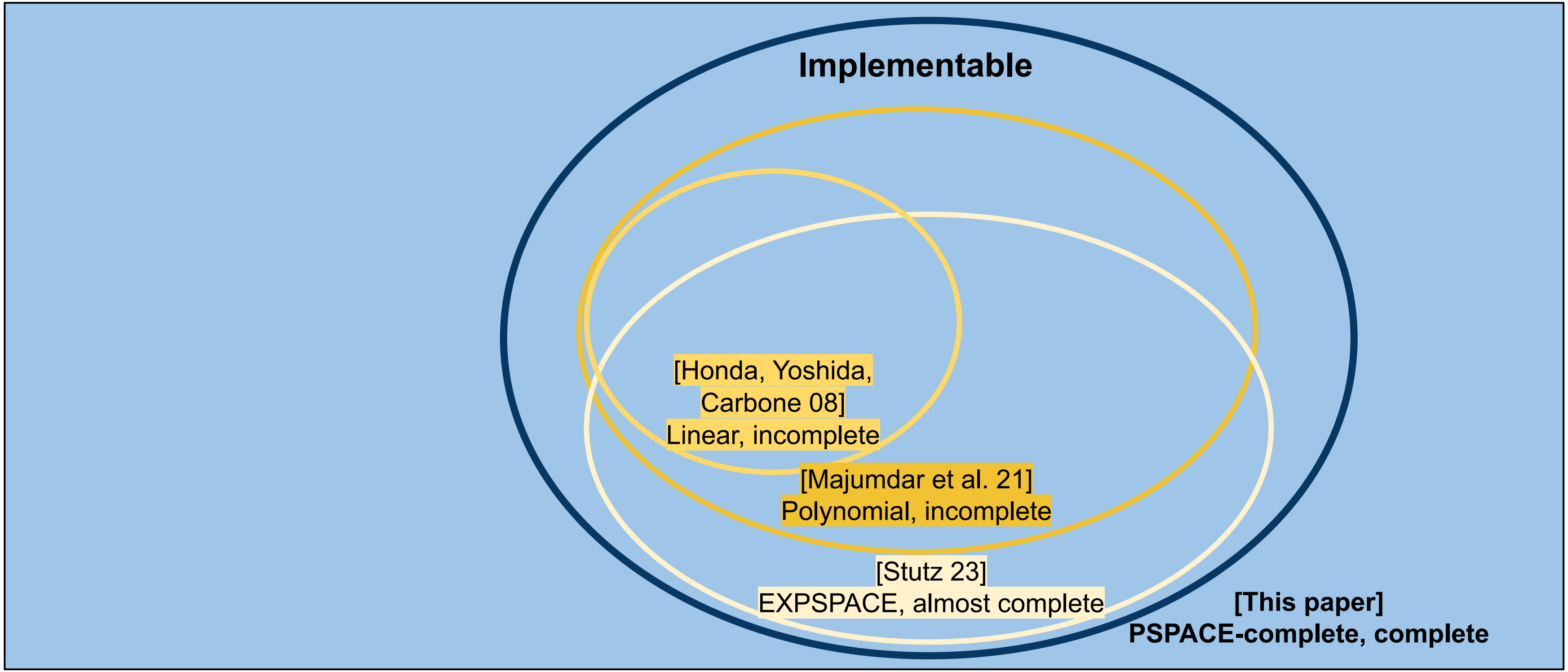
Theorem. The implementability problem for MSTs is PSPACE-complete.

Proof idea for lower bound: reduction to checking universality of NFAs.

Prototype evaluation

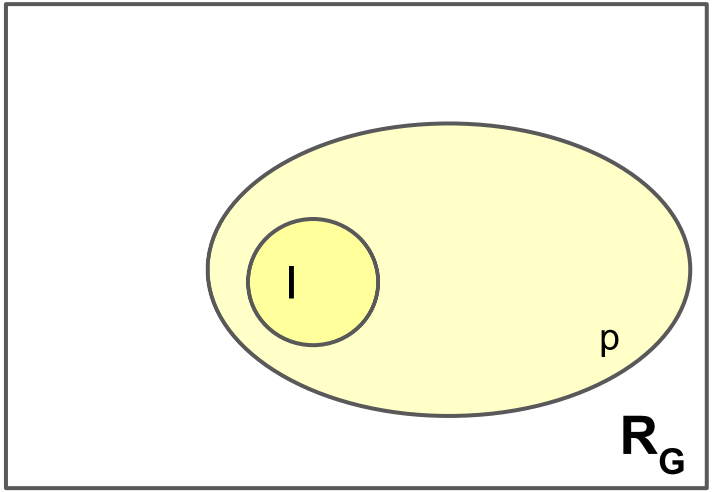


Thank you!


Implementable
[Honda, Yoshida, Carbone 08]
Linear, incomplete
[Majumdar et al. 21]
Polynomial, incomplete
[Stutz 23]
EXPSPACE, almost complete
[This paper]
PSPACE-complete, complete

Proof: Soundness (Projectable ⇒ Implementable)

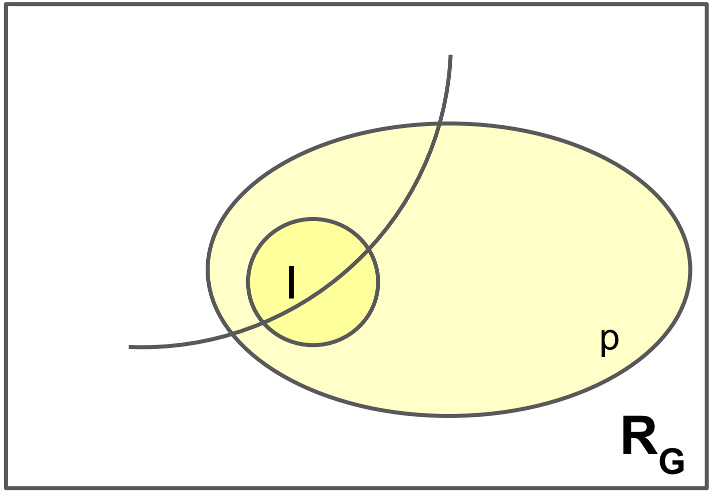
Send transitions shrink the intersection set in a principled way



trace = w

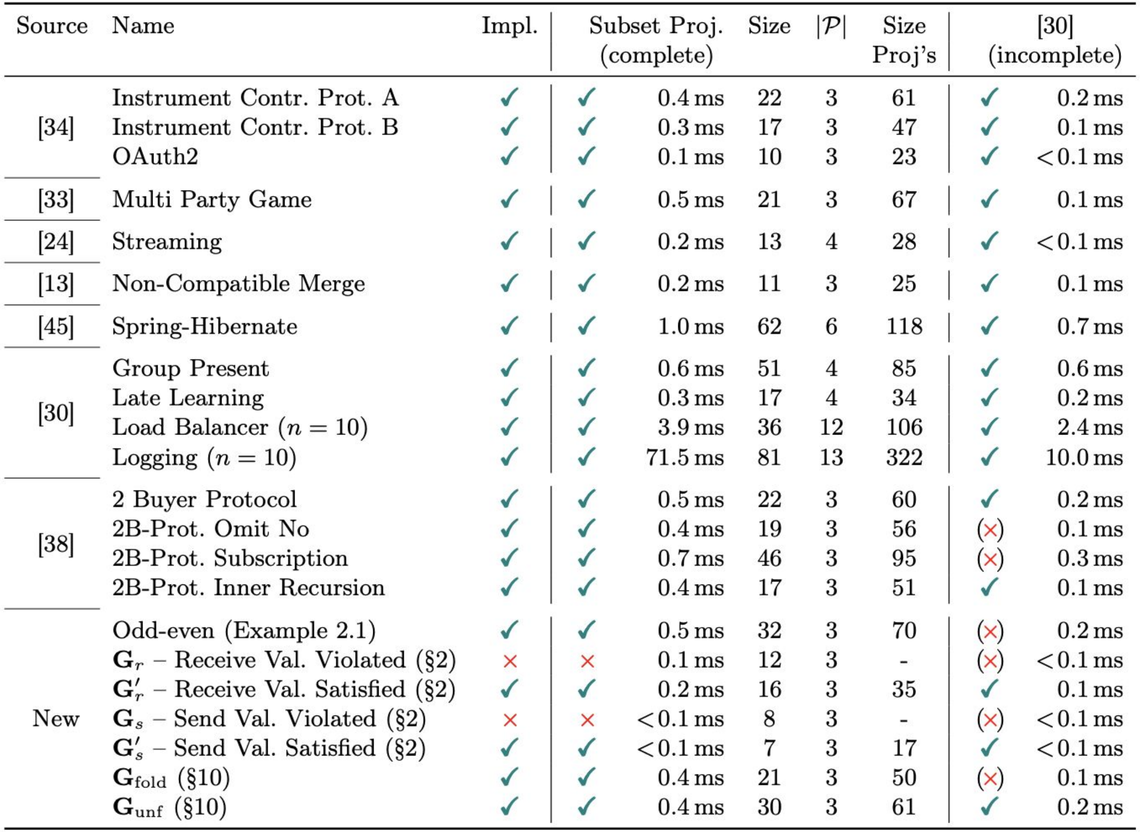
Proof: Soundness (Projectable ⇒ Implementable)

Send transitions shrink the intersection set in a principled way



trace = wx,
x is a send event for p

Prototype evaluation



Source	Name	Impl.	Subset Proj.		Size	$\lvert\mathcal{P}\rvert$	Size	[30]	
			(complete)				Proj's	(incomplete)	
[34]	Instrument Contr. Prot. A	✓	✓	0.4 ms	22	3	61	✓	0.2 ms
	Instrument Contr. Prot. B	✓	✓	0.3 ms	17	3	47	✓	0.1 ms
	OAuth2	✓	✓	0.1 ms	10	3	23	✓	< 0.1 ms
[33]	Multi Party Game	✓	✓	0.5 ms	21	3	67	✓	0.1 ms
[24]	Streaming	✓	✓	0.2 ms	13	4	28	✓	< 0.1 ms
[13]	Non-Compatible Merge	✓	✓	0.2 ms	11	3	25	✓	0.1 ms
[45]	Spring-Hibernate	✓	✓	1.0 ms	62	6	118	✓	0.7 ms
[30]	Group Present	✓	✓	0.6 ms	51	4	85	✓	0.6 ms
	Late Learning	✓	✓	0.3 ms	17	4	34	✓	0.2 ms
	Load Balancer ($n = 10$)	✓	✓	3.9 ms	36	12	106	✓	2.4 ms
	Logging ($n = 10$)	✓	✓	71.5 ms	81	13	322	✓	10.0 ms
[38]	2 Buyer Protocol	✓	✓	0.5 ms	22	3	60	✓	0.2 ms
	2B-Prot. Omit No	✓	✓	0.4 ms	19	3	56	(×)	0.1 ms
	2B-Prot. Subscription	✓	✓	0.7 ms	46	3	95	(×)	0.3 ms
	2B-Prot. Inner Recursion	✓	✓	0.4 ms	17	3	51	✓	0.1 ms
New	Odd-even (Example 2.1)	✓	✓	0.5 ms	32	3	70	(×)	0.2 ms
	$\mathbf{G}_r$ – Receive Val. Violated (§2)	×	×	0.1 ms	12	3	-	(×)	< 0.1 ms
	$\mathbf{G}'_r$ – Receive Val. Satisfied (§2)	✓	✓	0.2 ms	16	3	35	✓	0.1 ms
	$\mathbf{G}_s$ – Send Val. Violated (§2)	×	×	< 0.1 ms	8	3	-	(×)	< 0.1 ms
	$\mathbf{G}'_s$ – Send Val. Satisfied (§2)	✓	✓	< 0.1 ms	7	3	17	✓	< 0.1 ms
	$\mathbf{G}_{\text{fold}}$ (§10)	✓	✓	0.4 ms	21	3	50	(×)	0.1 ms
	$\mathbf{G}_{\text{unf}}$ (§10)	✓	✓	0.4 ms	30	3	61	✓	0.2 ms