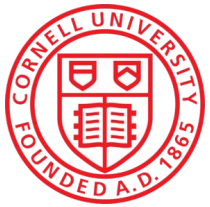
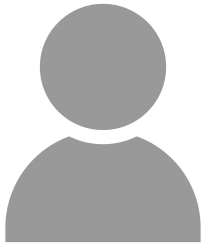
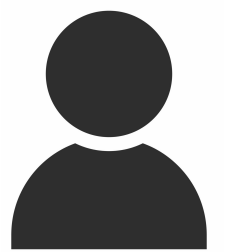
Implementability of Global Protocols modulo Network Architectures

Elaine Li Thomas Wies

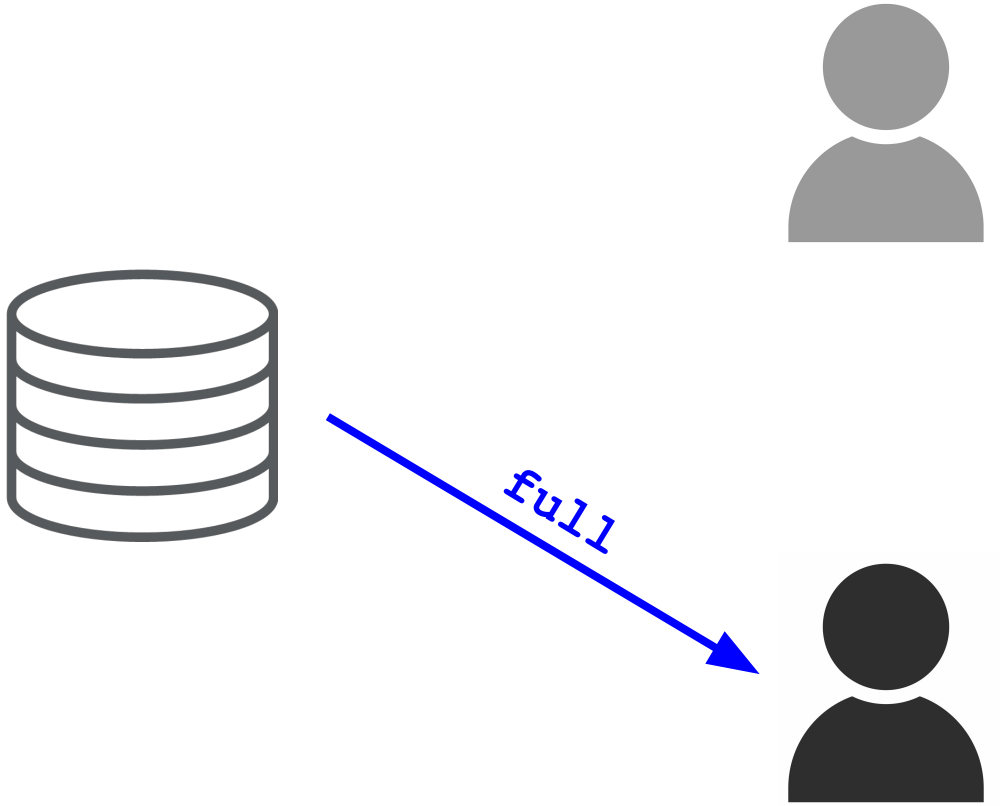


Task delegation protocol

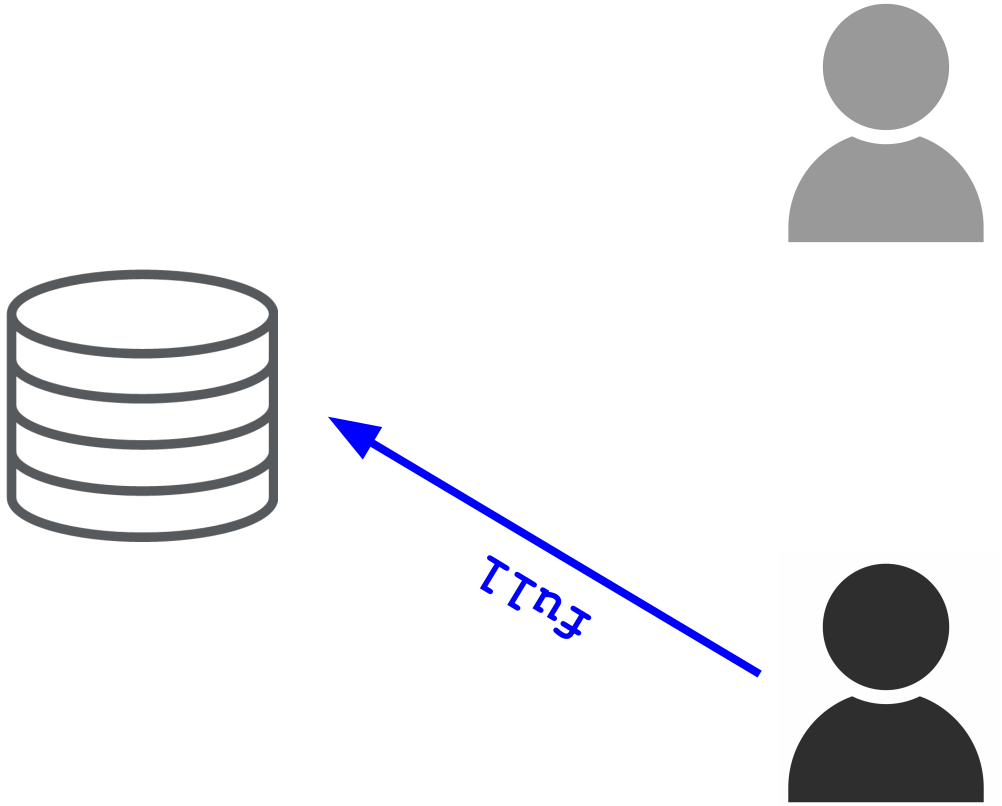




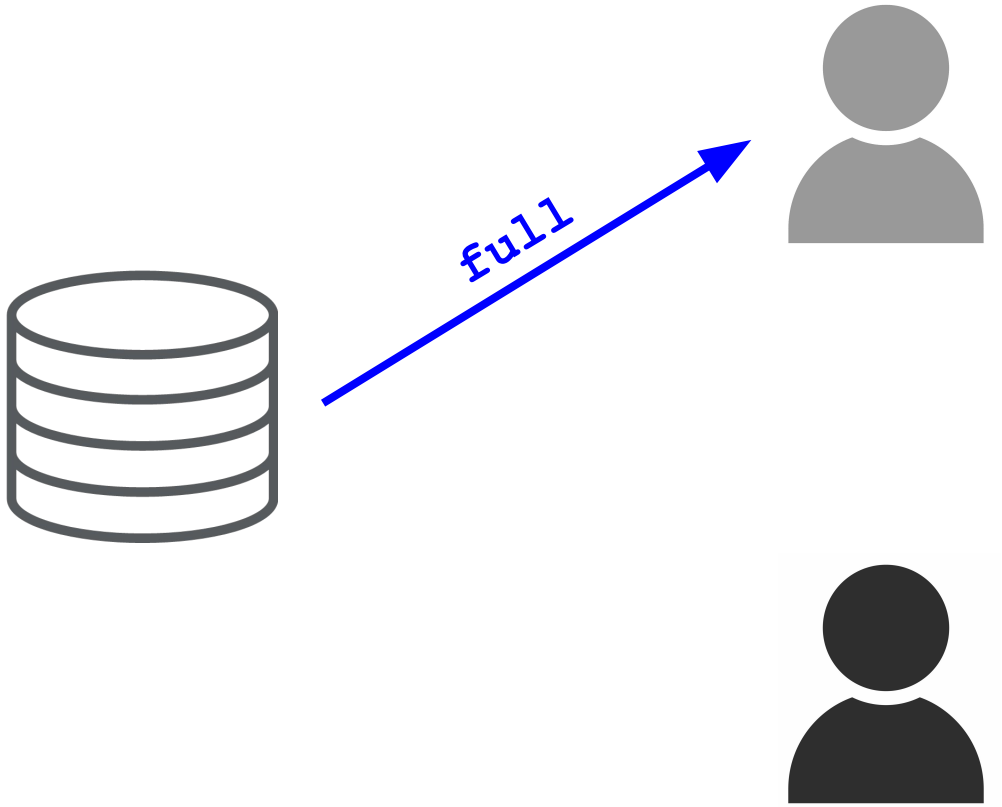
Task delegation protocol



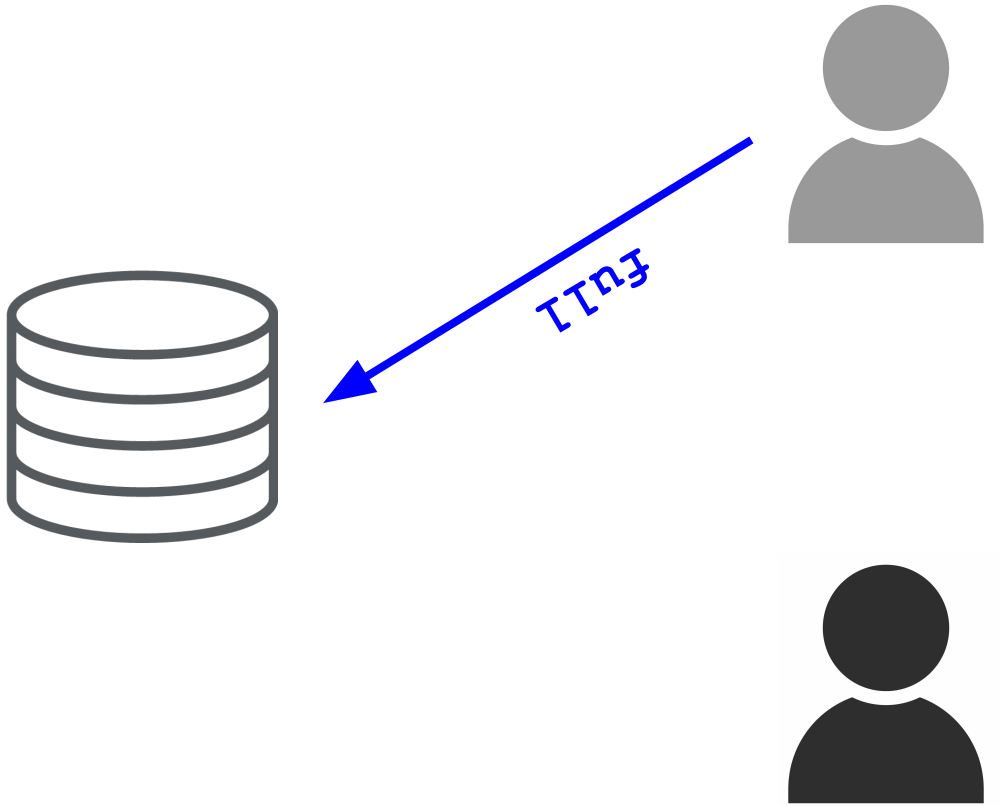
Task delegation protocol



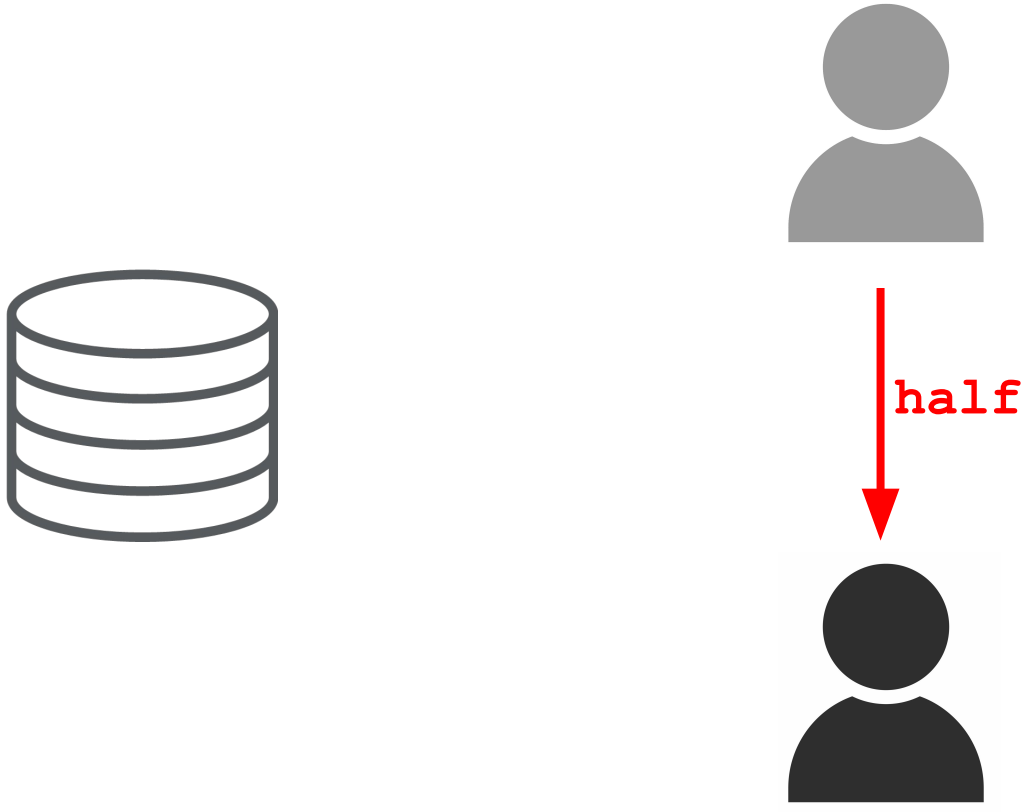
Task delegation protocol



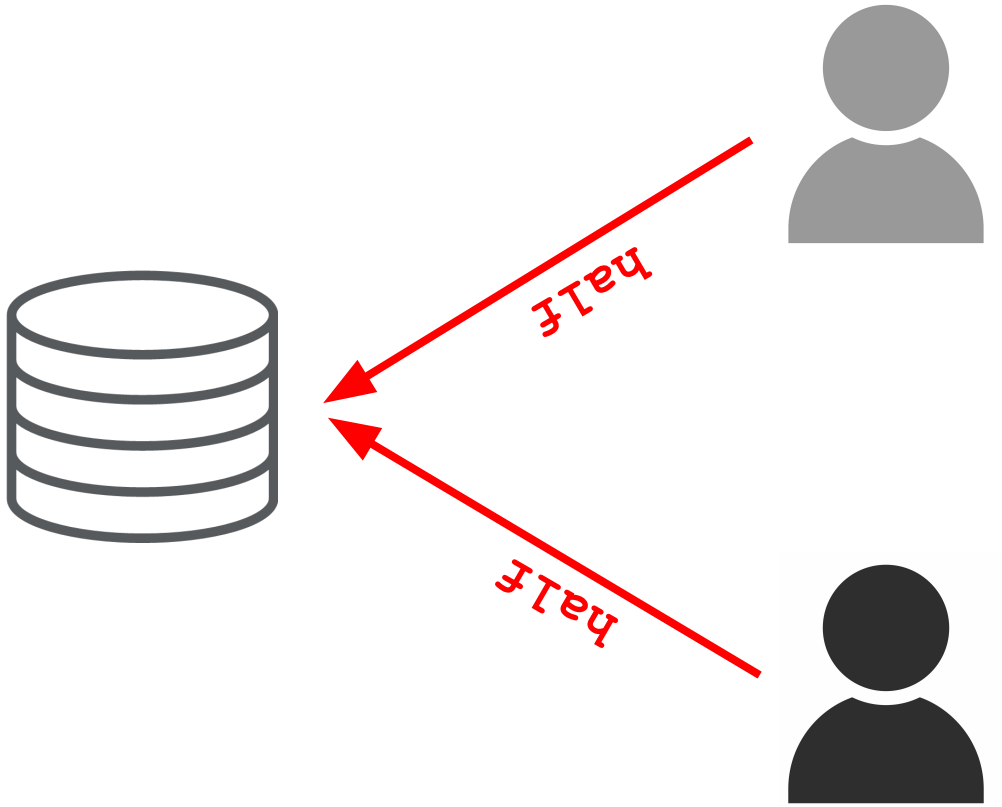
Task delegation protocol



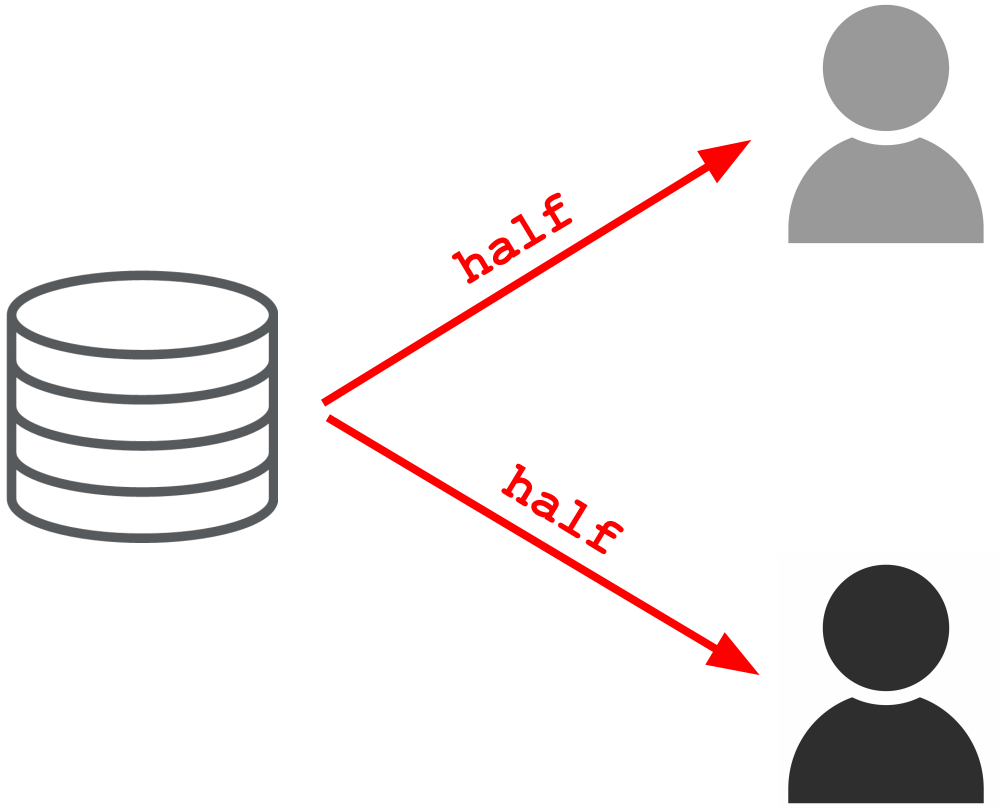
Task delegation protocol



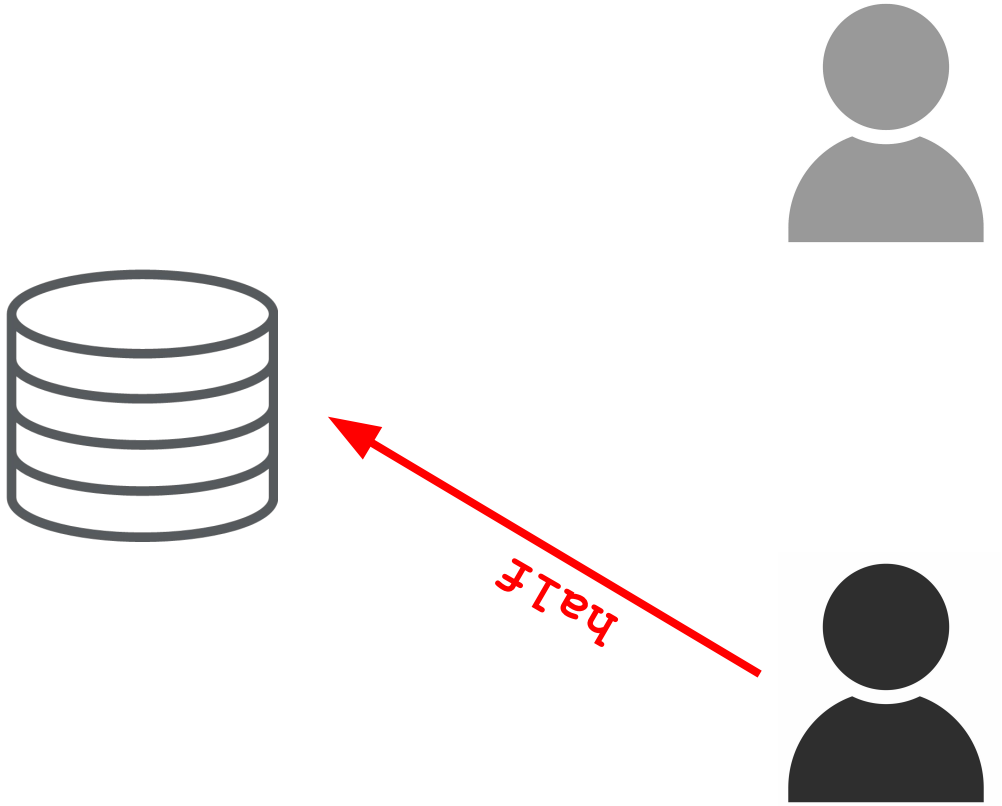
Task delegation protocol



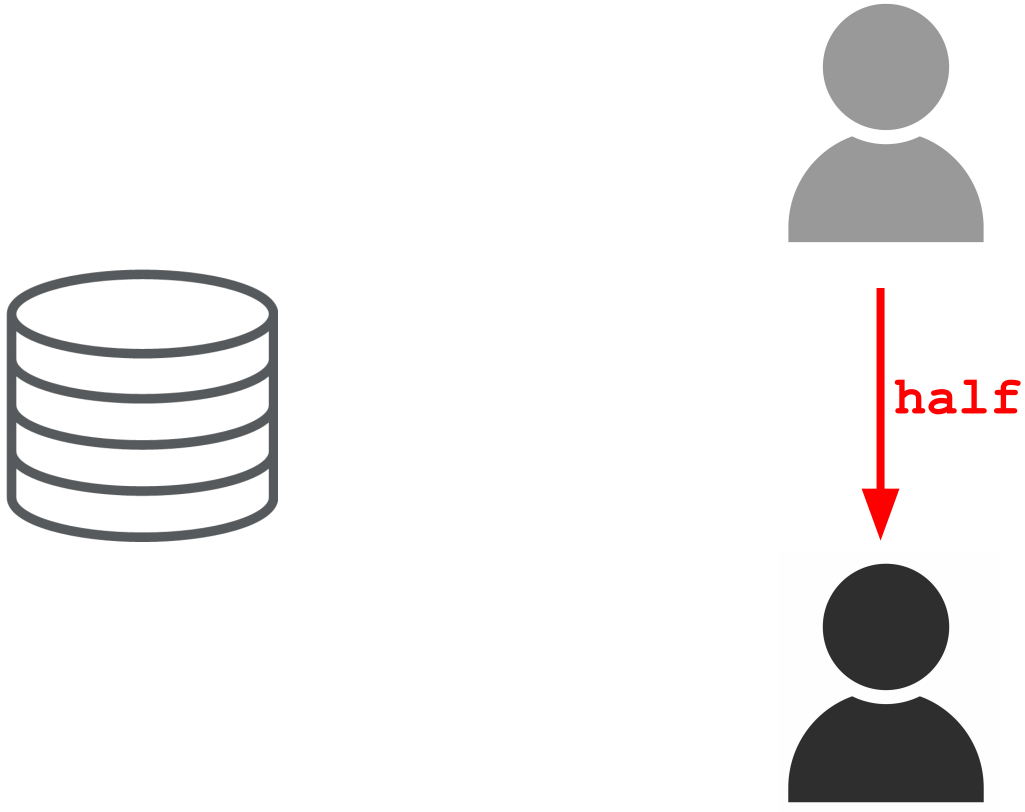
Task delegation protocol



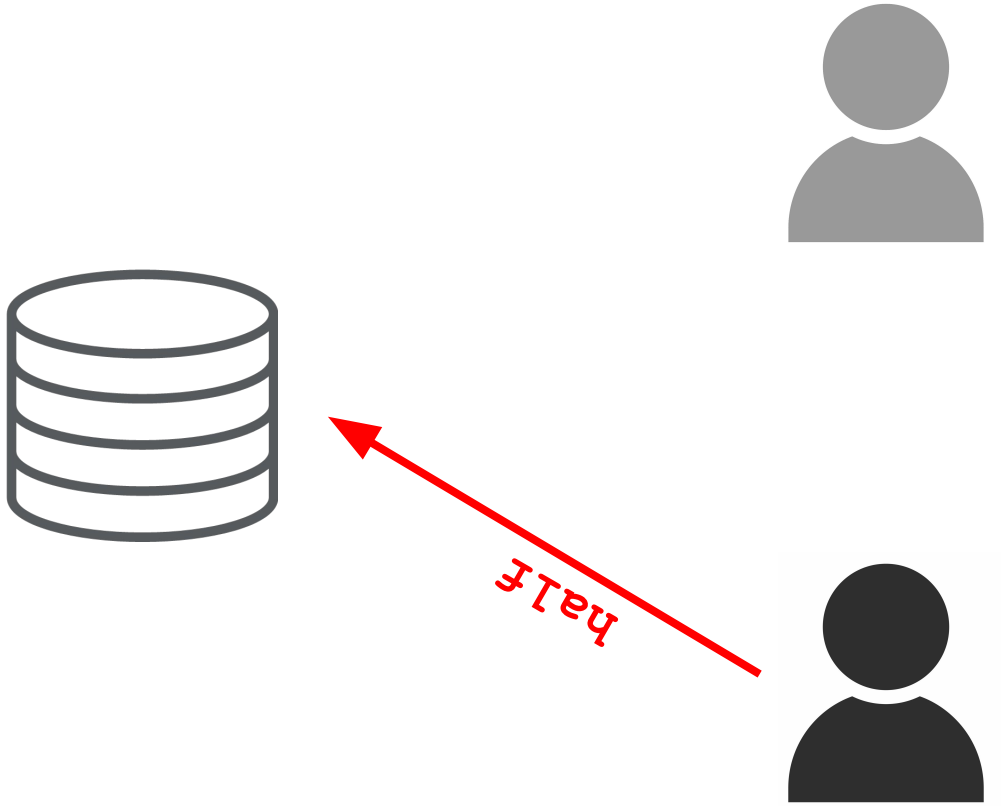
Task delegation protocol



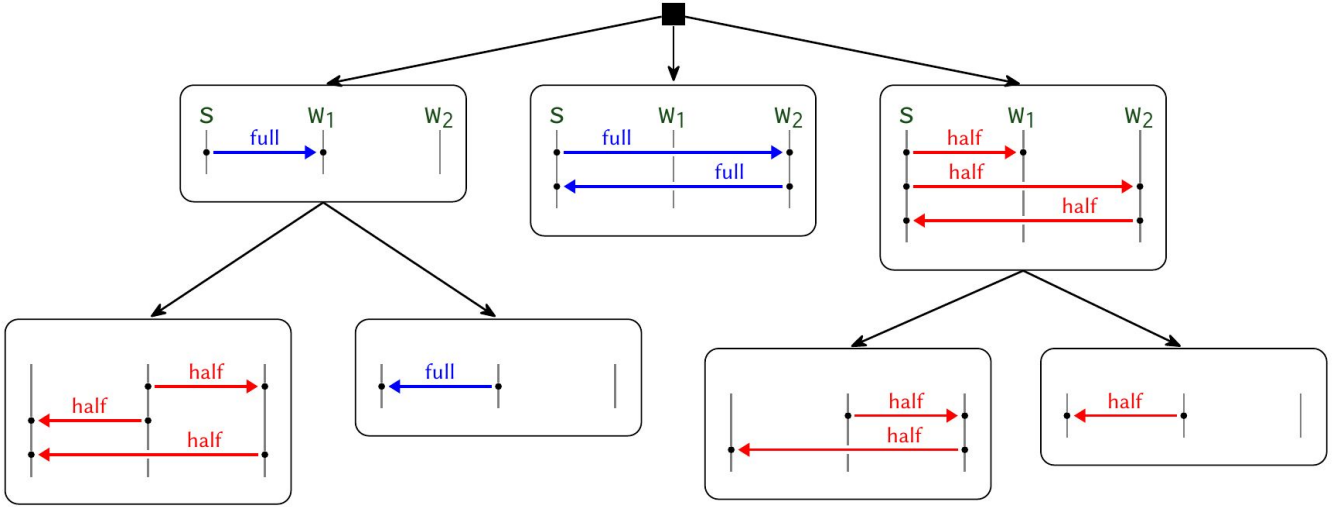
Task delegation protocol



Task delegation protocol

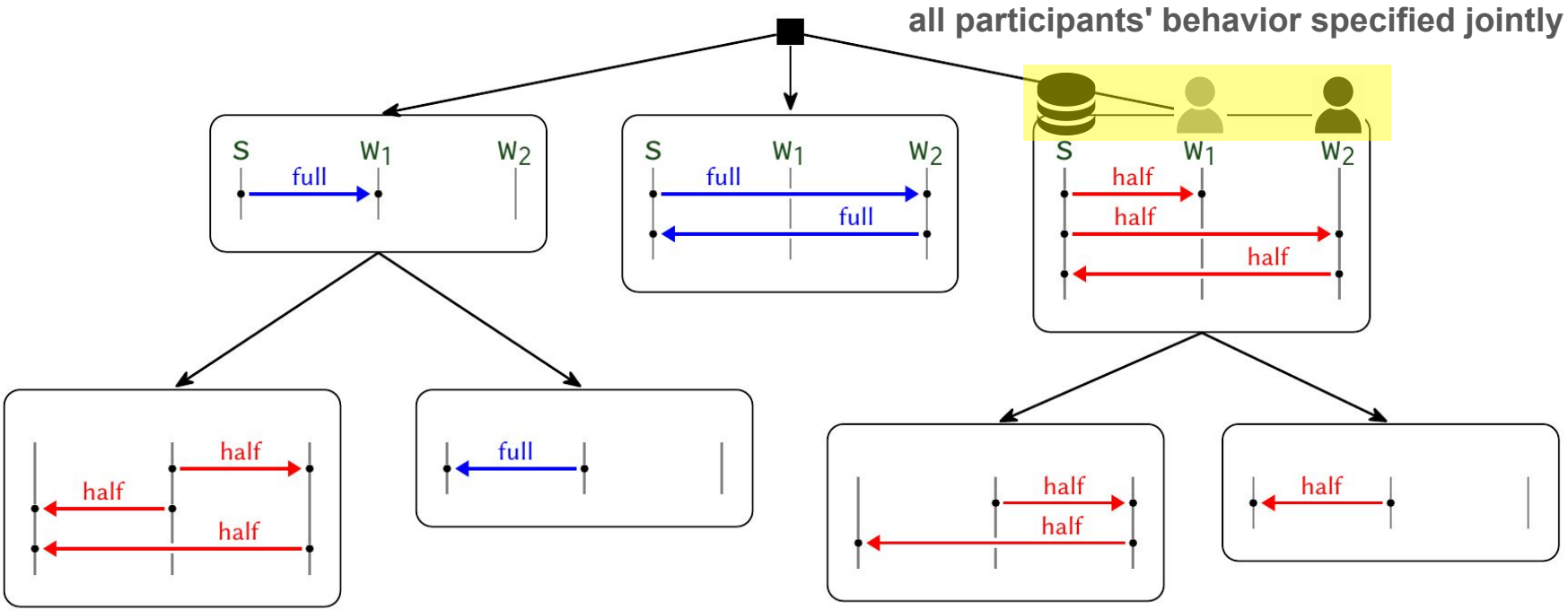


Global protocols

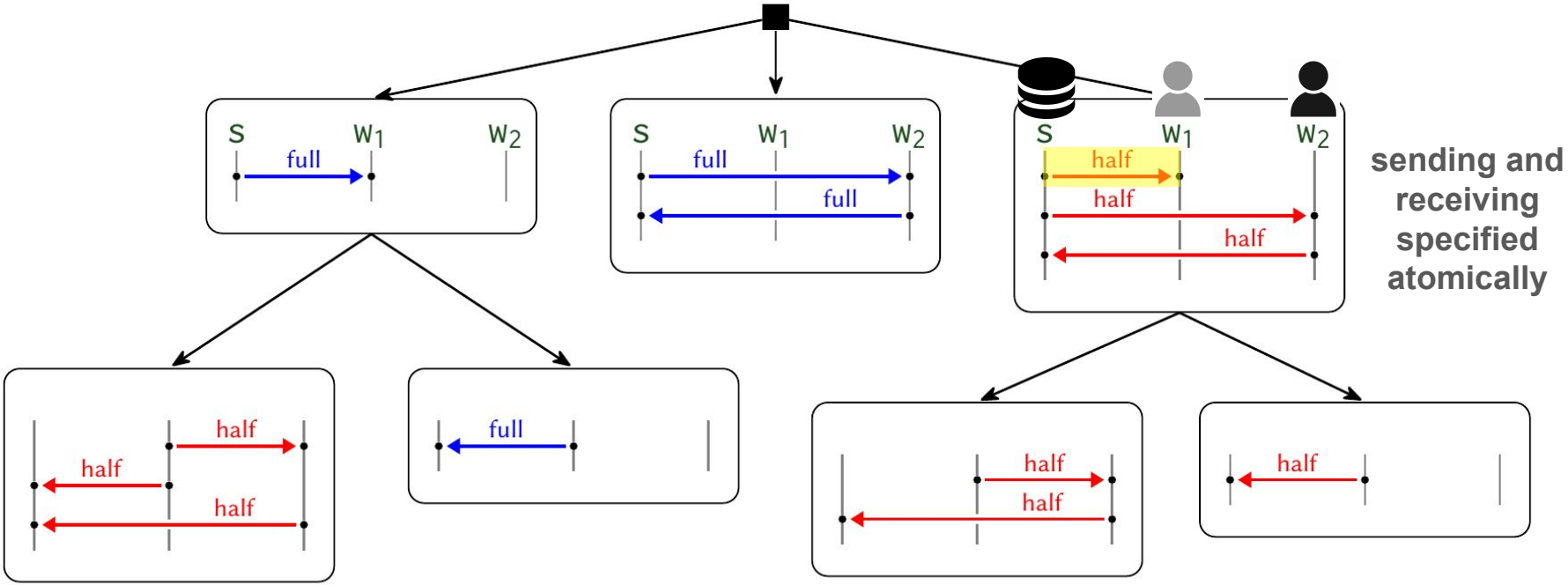


- High-level message sequence charts [Mauw and Reniers 97]
- Session types [Honda et al. 08]
- Choreographic programs [Carbone and Montesi 13]

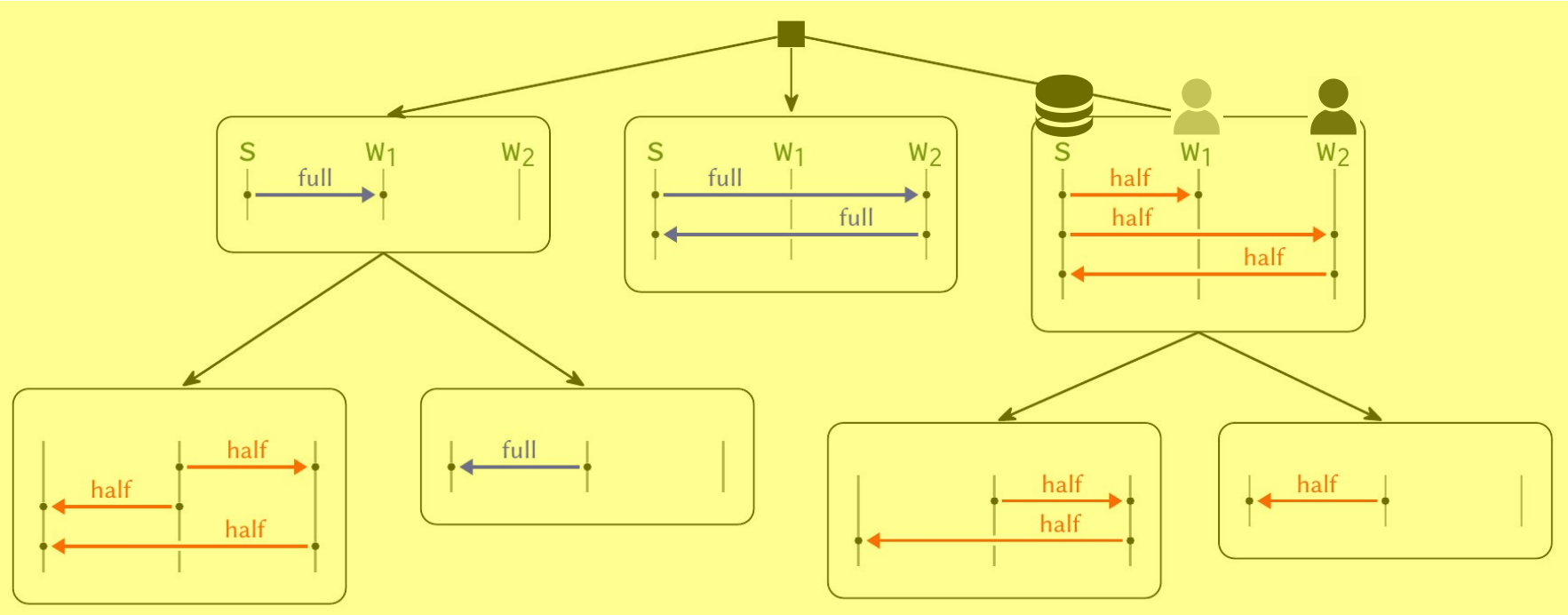
Global protocols



Global protocols

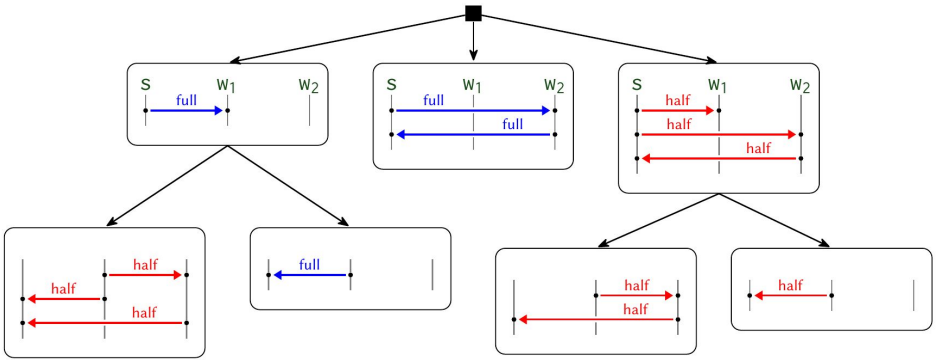


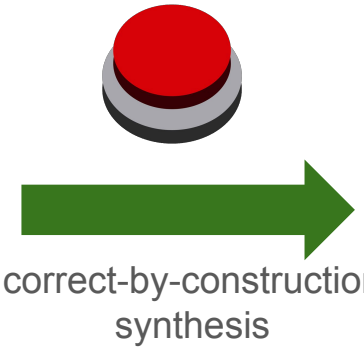
Global protocols

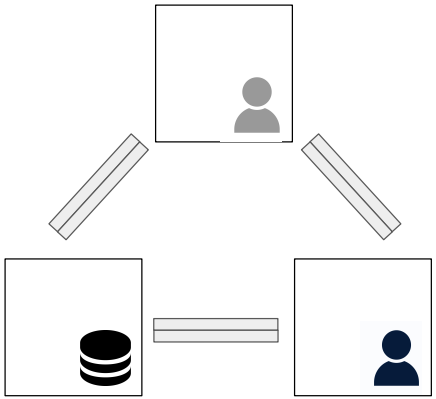


global control flow

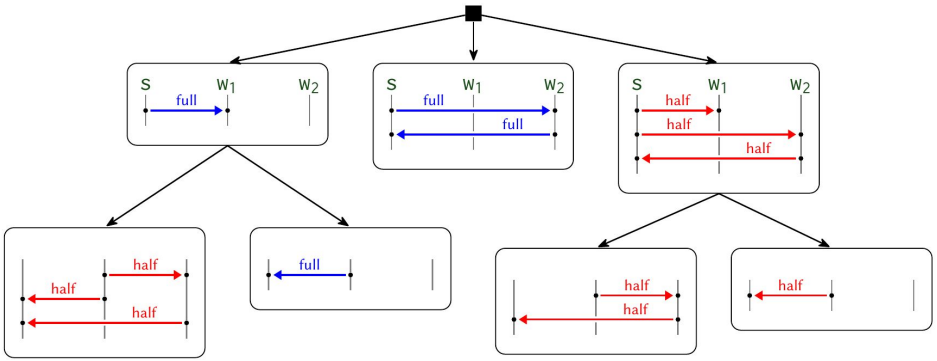
Top-down verification







Top-down verification

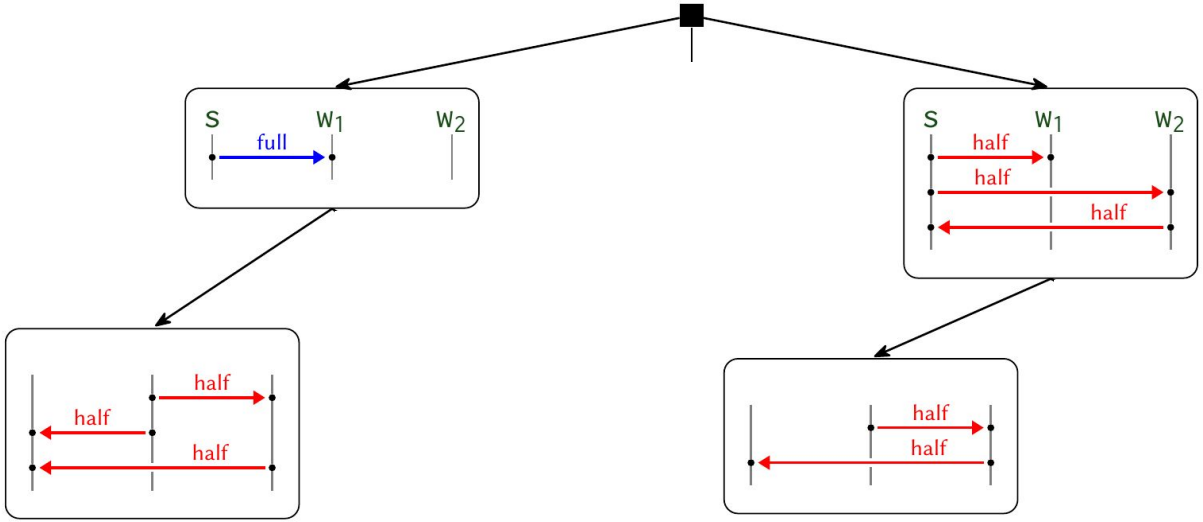


no message channels, omniscient coordinator etc.

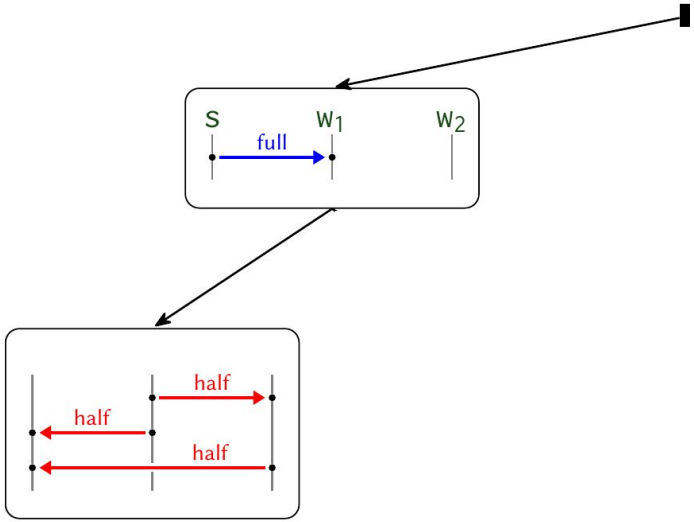
?

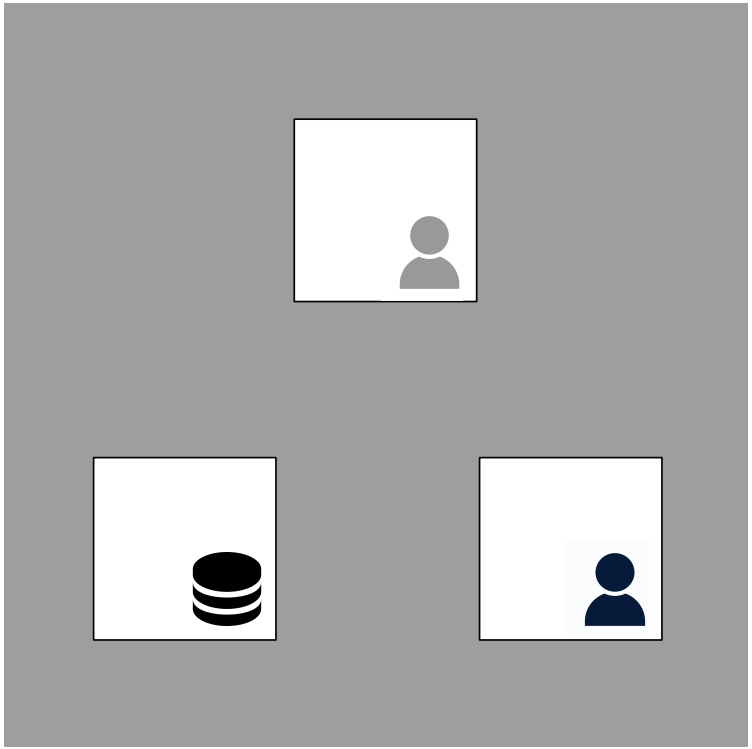
realizability/
implementability
[Alur et al. 00]

Non-implementability

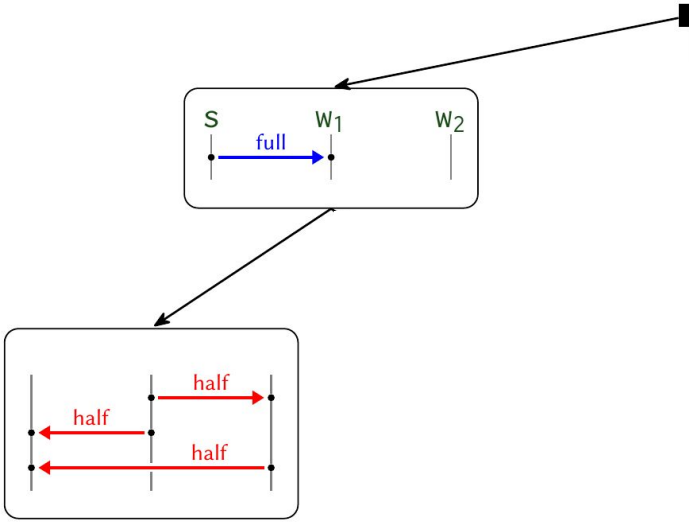


Non-implementability

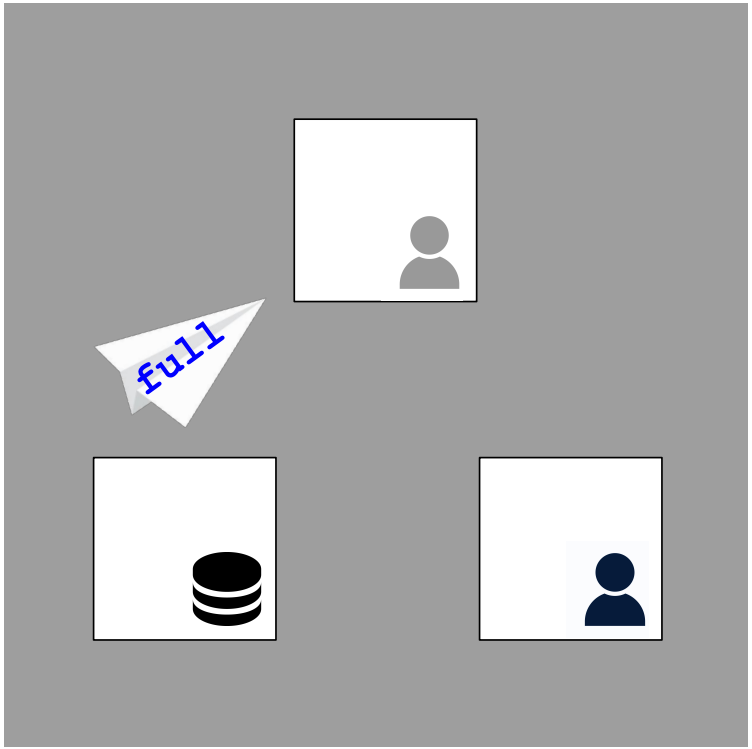




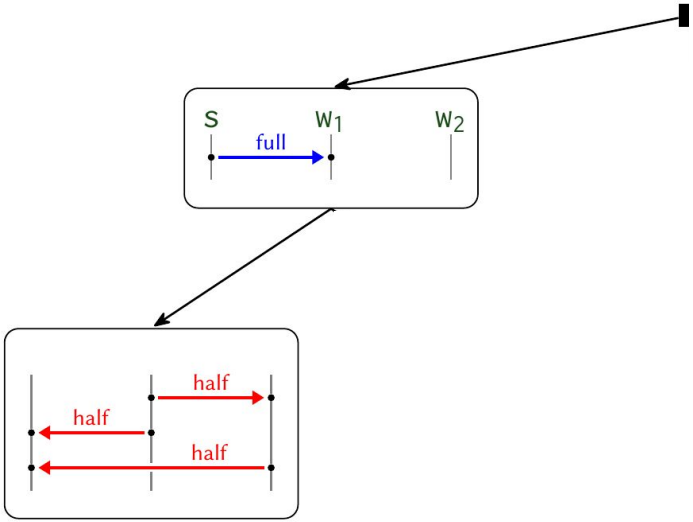
Non-implementability



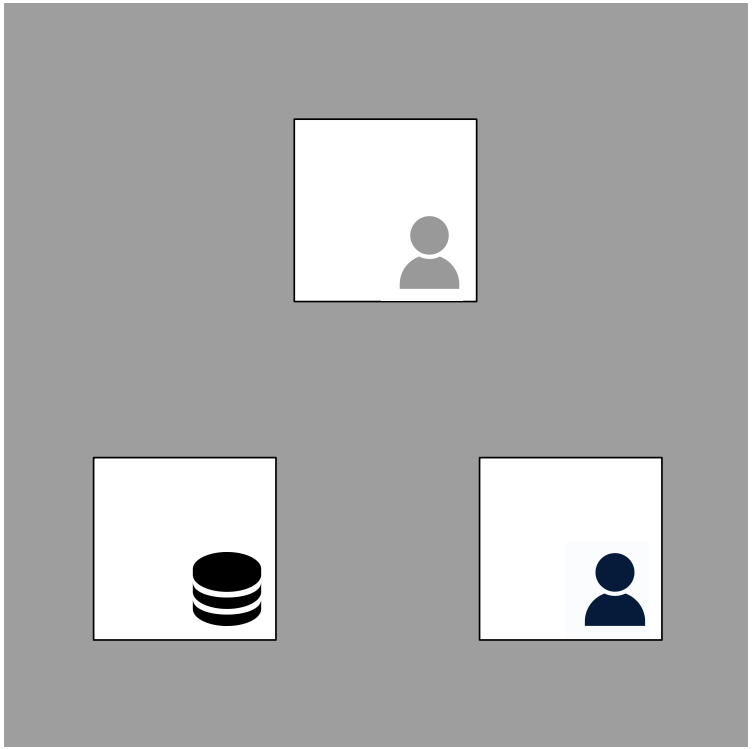
$u_1 = s \triangleright w_1 ! full$



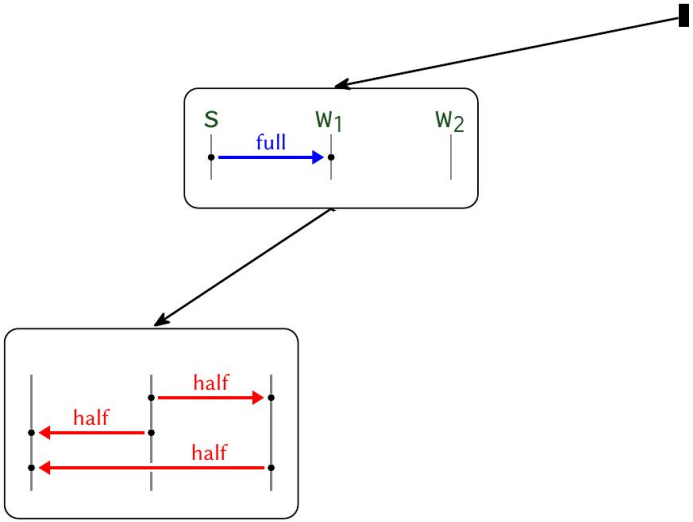
Non-implementability



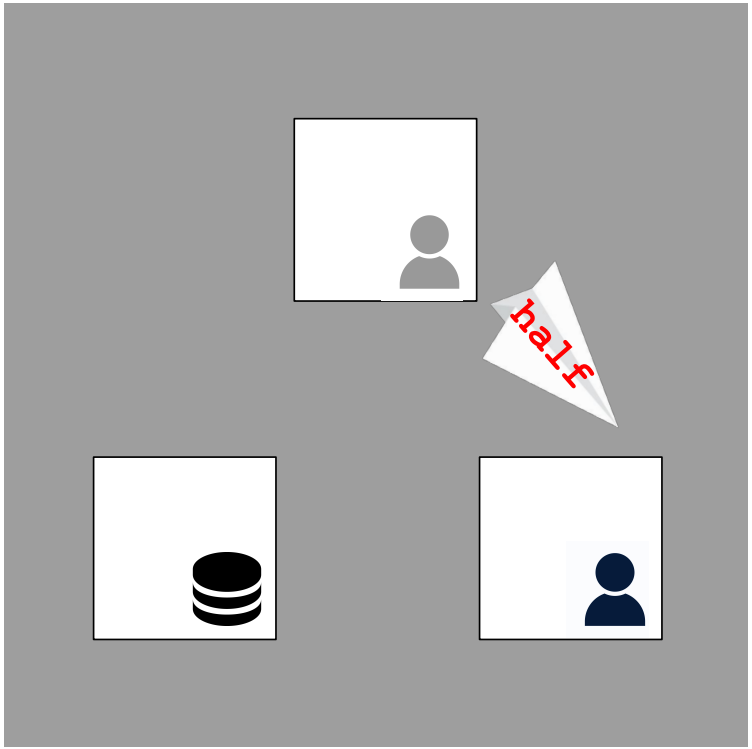
$u_1 = \mathsf{S} \triangleright \mathsf{W}_1!\mathsf{full} \cdot \mathsf{W}_1 \triangleleft \mathsf{S}?\mathsf{full}$



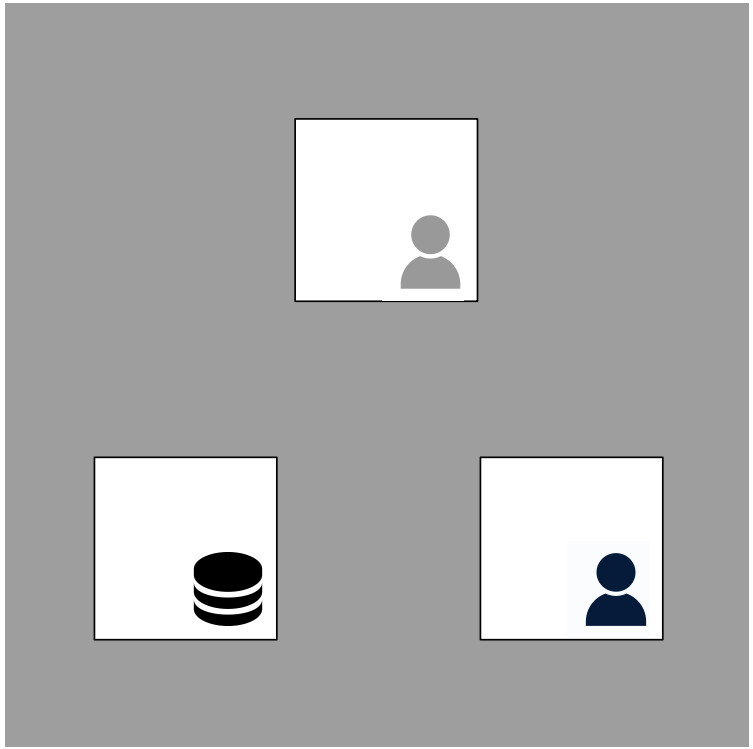
Non-implementability

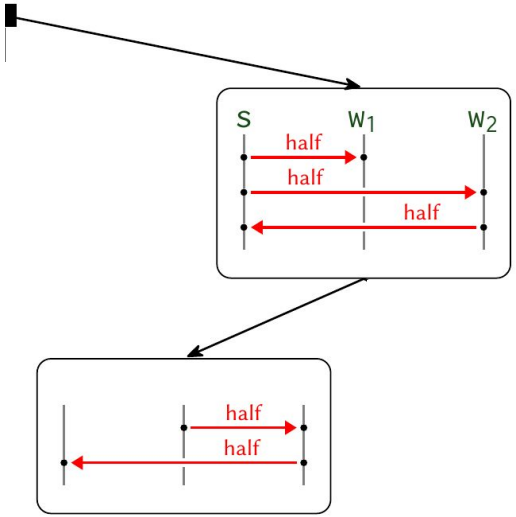


$u_1 = \mathsf{s} \triangleright \mathsf{w}_1!\mathsf{full} \cdot \mathsf{w}_1 \triangleleft \mathsf{s}?\mathsf{full} \cdot \mathsf{w}_1 \triangleright \mathsf{w}_2!\mathsf{half}$

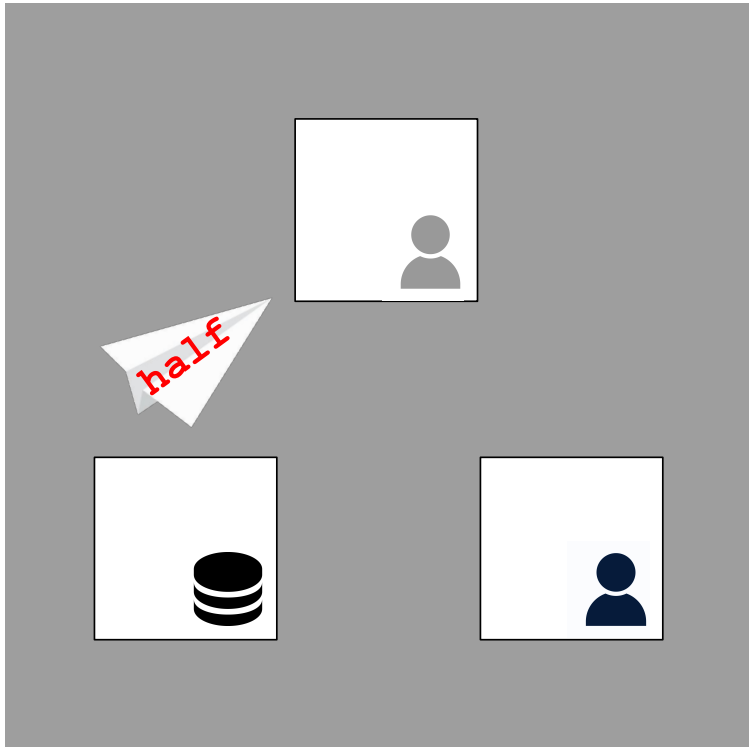


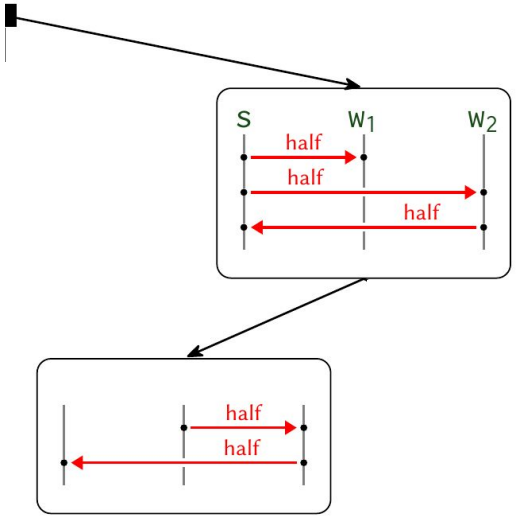
Non-implementability





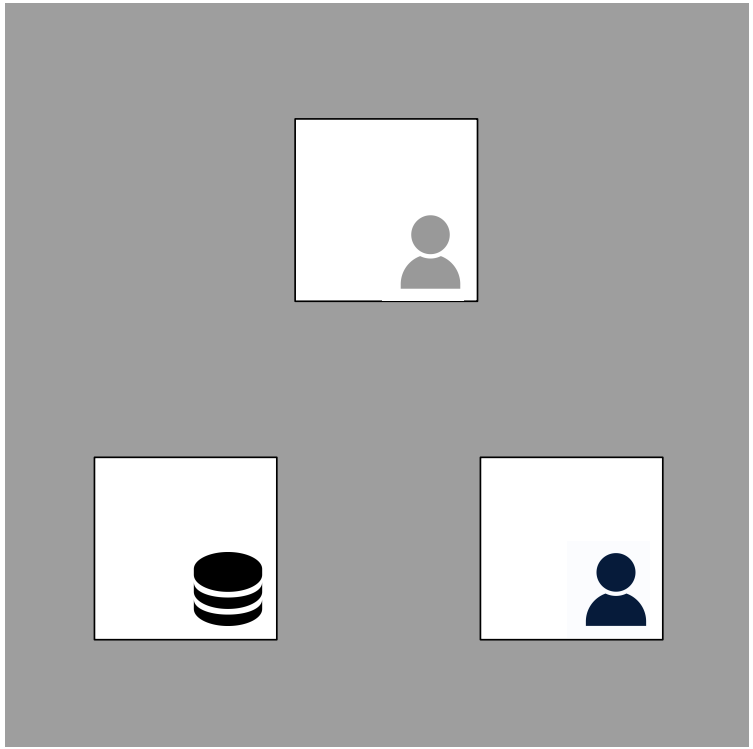
Non-implementability

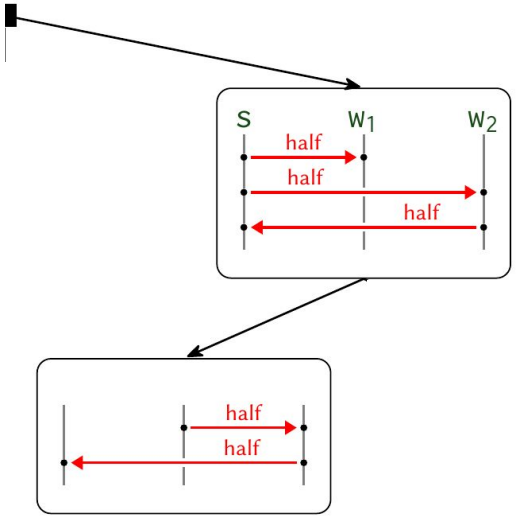




$u_2 = \mathsf{S} \triangleright \mathsf{W}_1!\mathsf{half}$

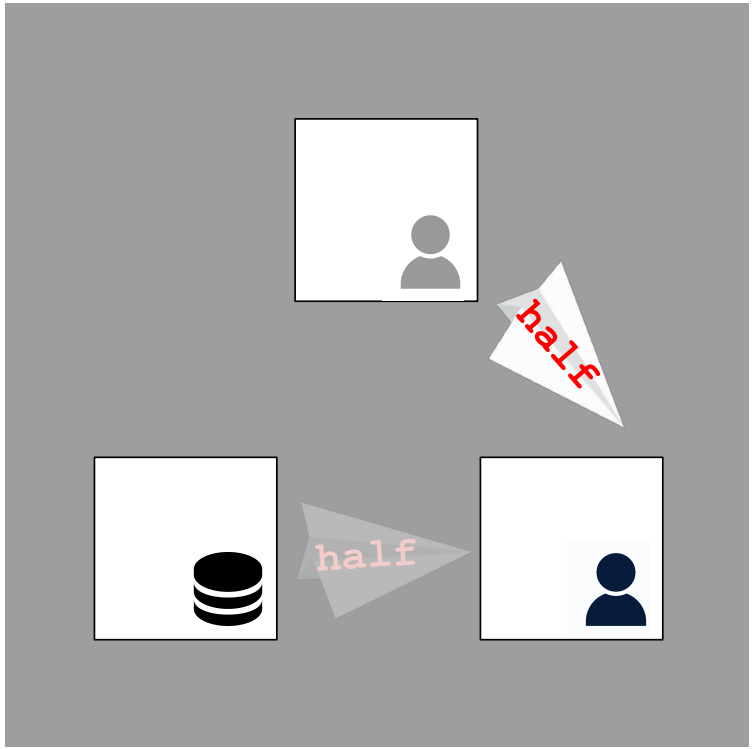
Non-implementability

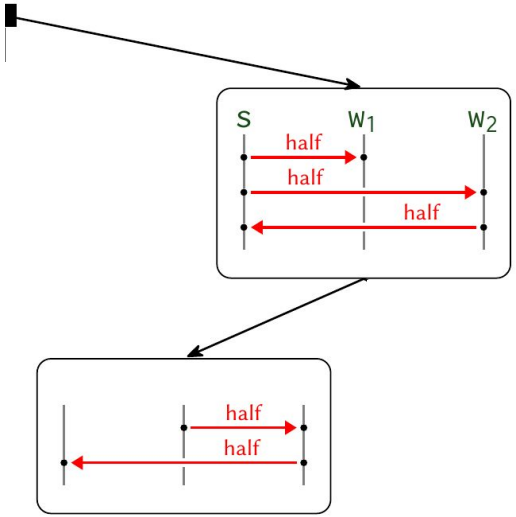




$u_2 = \mathsf{s} \triangleright \mathsf{w}_1!\mathsf{half} \cdot \mathsf{w}_1 \triangleleft \mathsf{s}?\mathsf{half}$

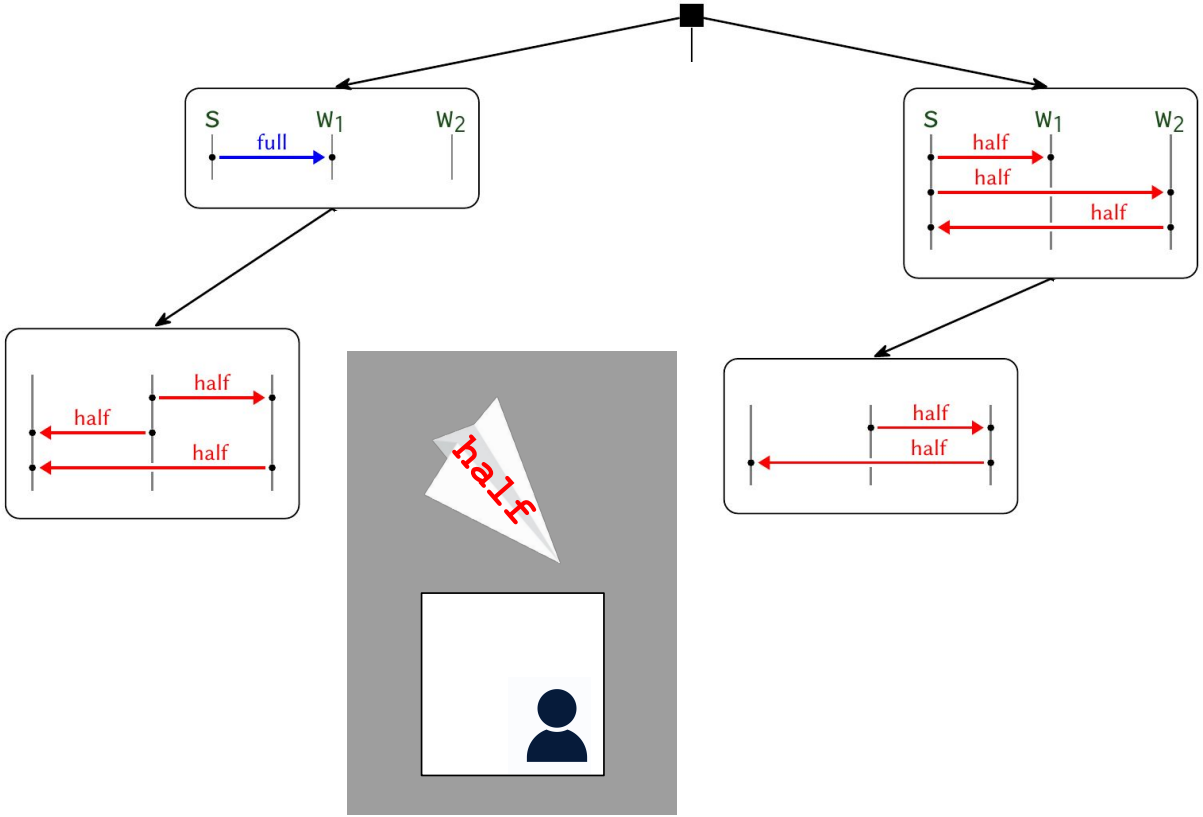
Non-implementability



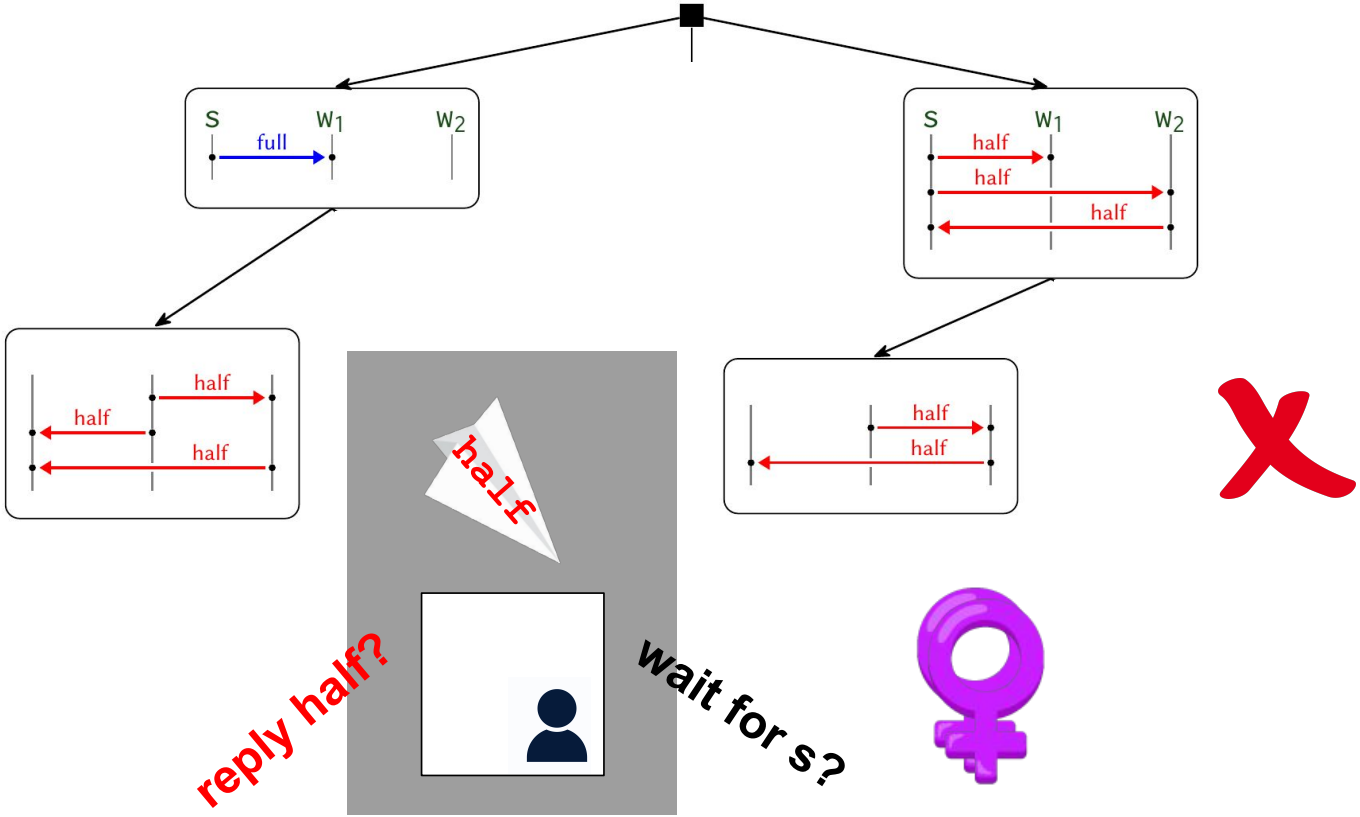


$$u_2 = \mathsf{S} \triangleright \mathsf{W}_1!\mathsf{half} \cdot \mathsf{W}_1 \triangleleft \mathsf{S}?\mathsf{half} \cdot \mathsf{W}_1 \triangleright \mathsf{W}_2!\mathsf{half}$$

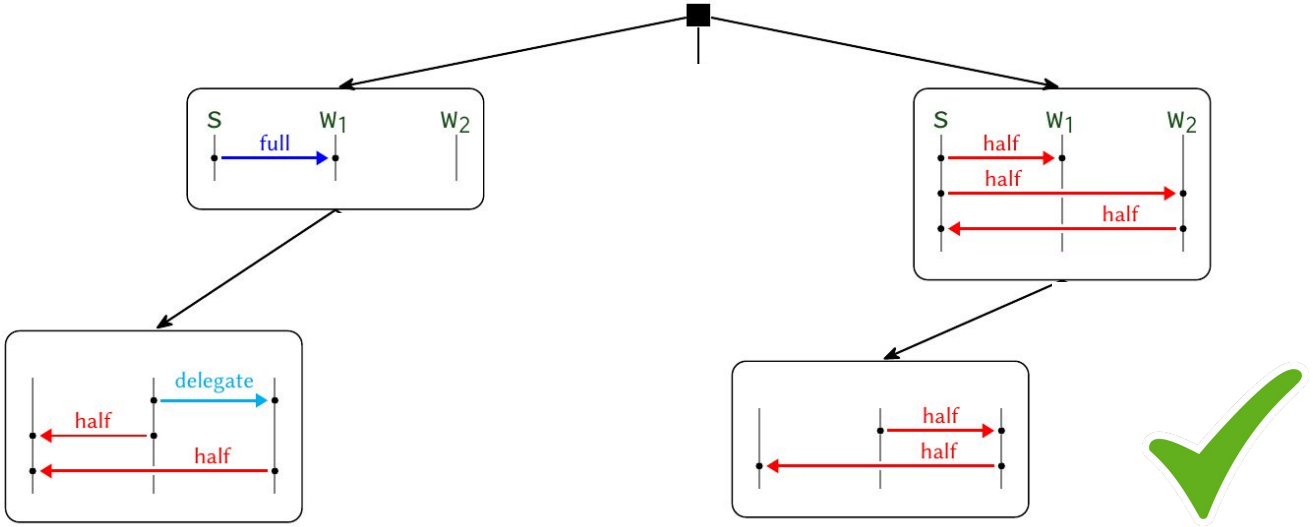
Non-implementability



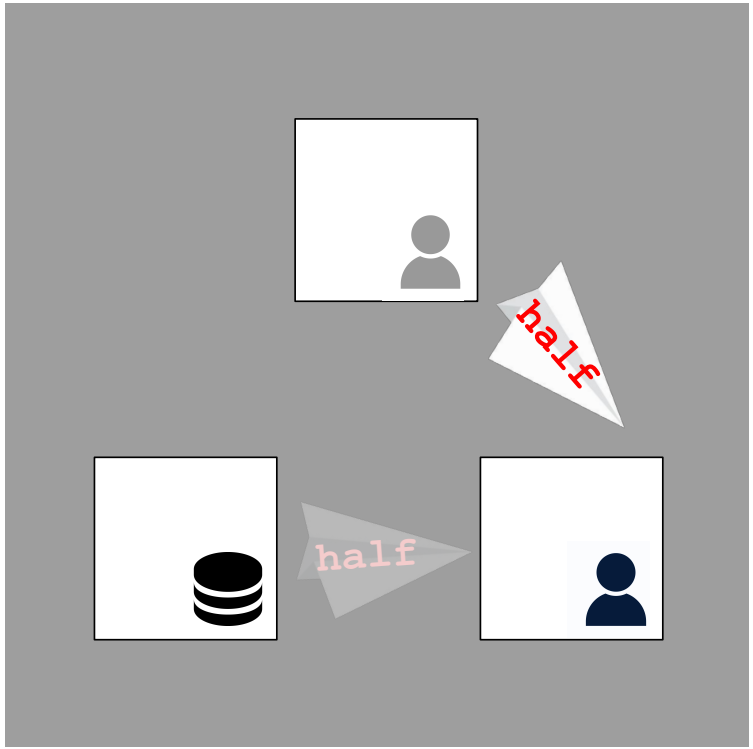
Non-implementability

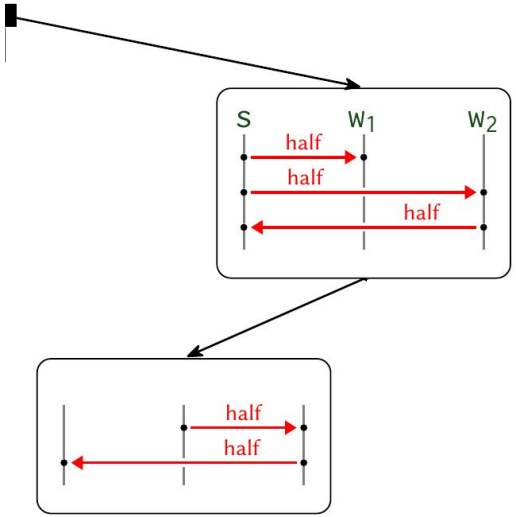


Non-implementability



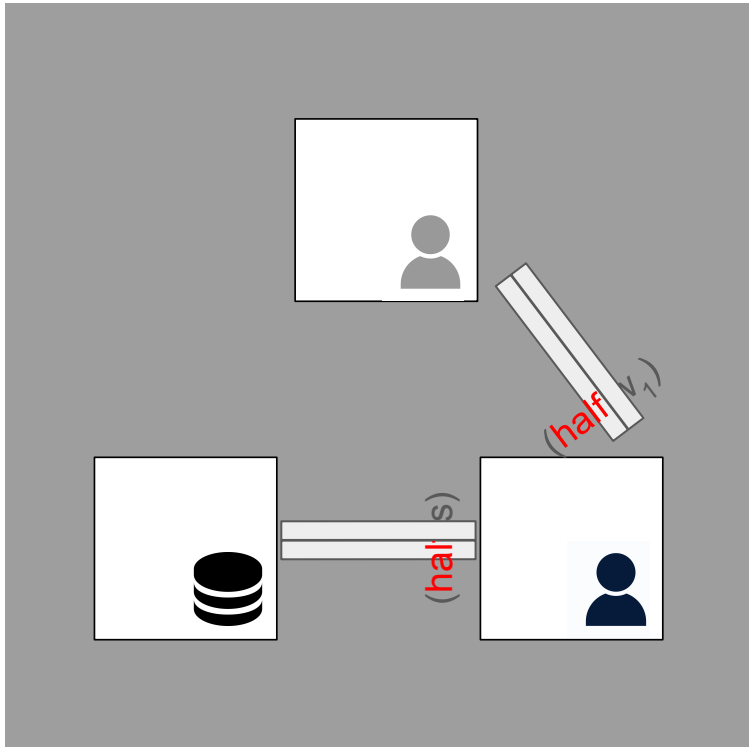
Non-implementability

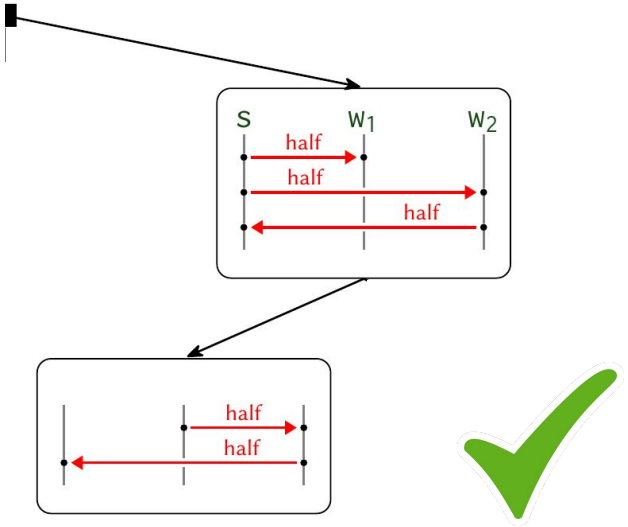




$$u_3 = \mathsf{s} \triangleright \mathsf{w}_1!\mathsf{half} \cdot \mathsf{w}_1 \triangleleft \mathsf{s}?\mathsf{half} \cdot \mathsf{w}_1 \triangleright \mathsf{w}_2!\mathsf{half} \cdot \mathsf{s} \triangleright \mathsf{w}_2!\mathsf{half}$$

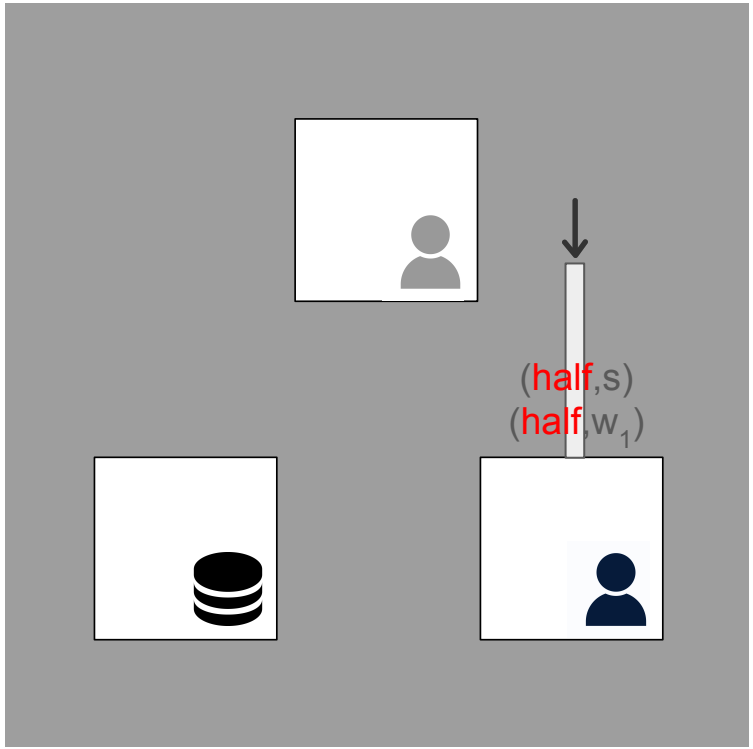
Non-implementability

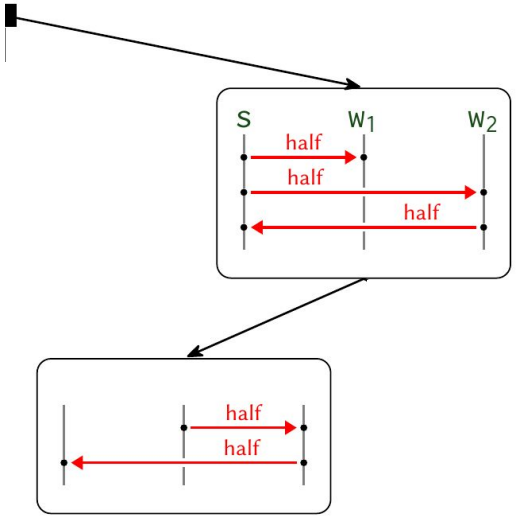




$u_3 = \mathsf{s} \triangleright \mathsf{w}_1!\mathsf{half} \cdot \mathsf{w}_1 \triangleleft \mathsf{s}?\mathsf{half} \cdot \mathsf{w}_1 \triangleright \mathsf{w}_2!\mathsf{half} \cdot \mathsf{s} \triangleright \mathsf{w}_2!\mathsf{half}$

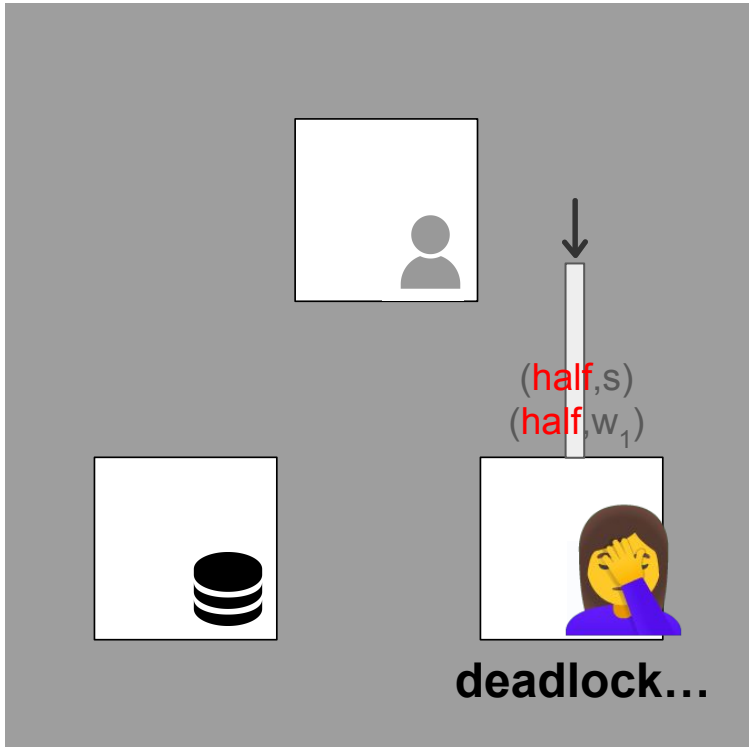
Non-implementability

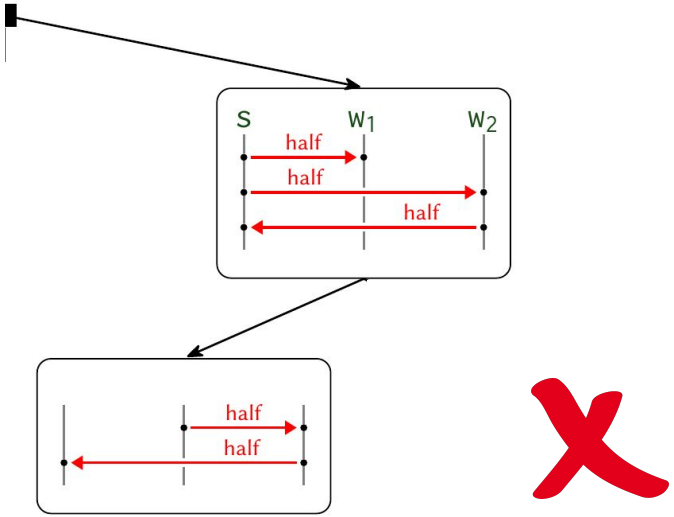




$$u_3 = s \triangleright w_1!\mathrm{half} \cdot w_1 \triangleleft s?\mathrm{half} \cdot w_1 \triangleright w_2!\mathrm{half} \cdot s \triangleright w_2!\mathrm{half}$$

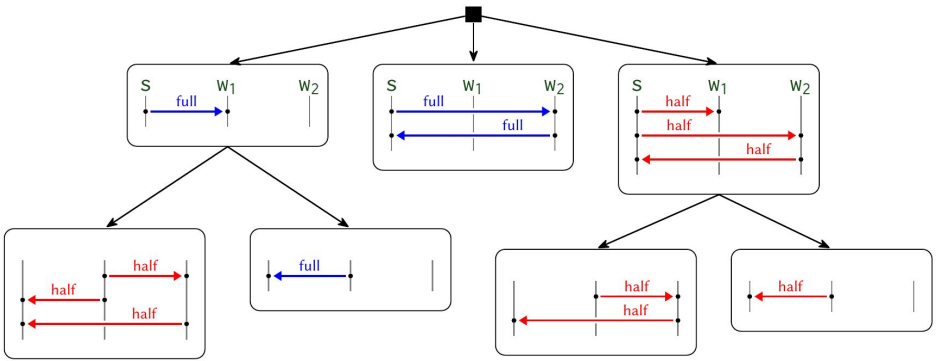
Non-implementability





$u_3 = s \triangleright w_1!\text{half} \cdot w_1 \triangleleft s?\text{half} \cdot w_1 \triangleright w_2!\text{half} \cdot s \triangleright w_2!\text{half}$

Network-parametric implementability



?

realizability/
implementability
under a given network?

Network-parametric goal

A single implementability characterization
and proof of preciseness
for all asynchronous networks

Network-parametric goal

A single implementability characterization
and proof of preciseness
for all **asynchronous networks**

"On the diversity of asynchronous communication" [Chevrou et al. 16]

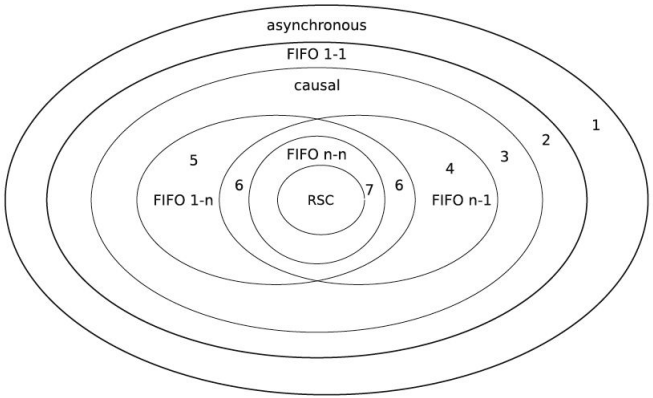
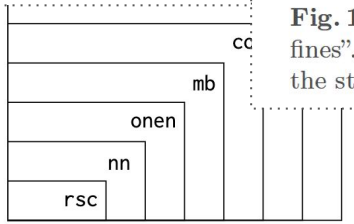


Fig. 7. Hierarchy of the communication models



(a) The hierarchy of MSC classes.

[Di Giusto et al. 23]

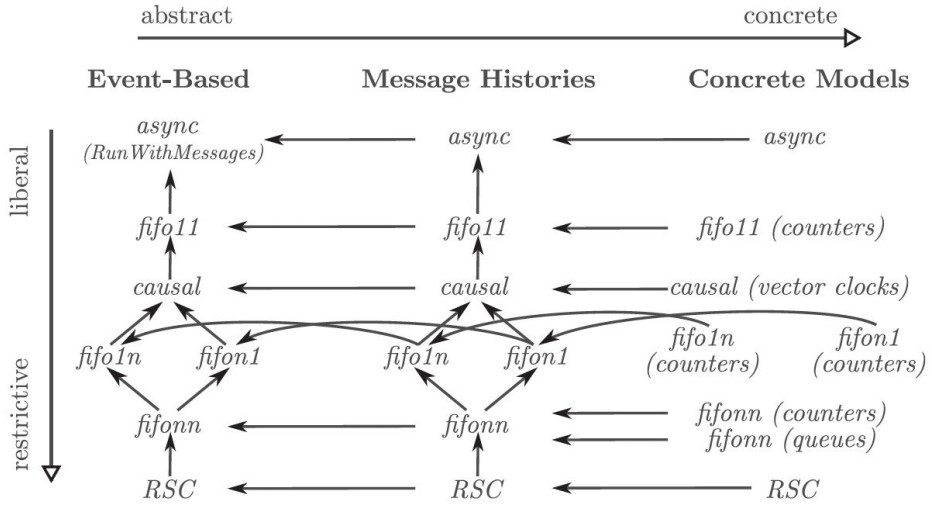
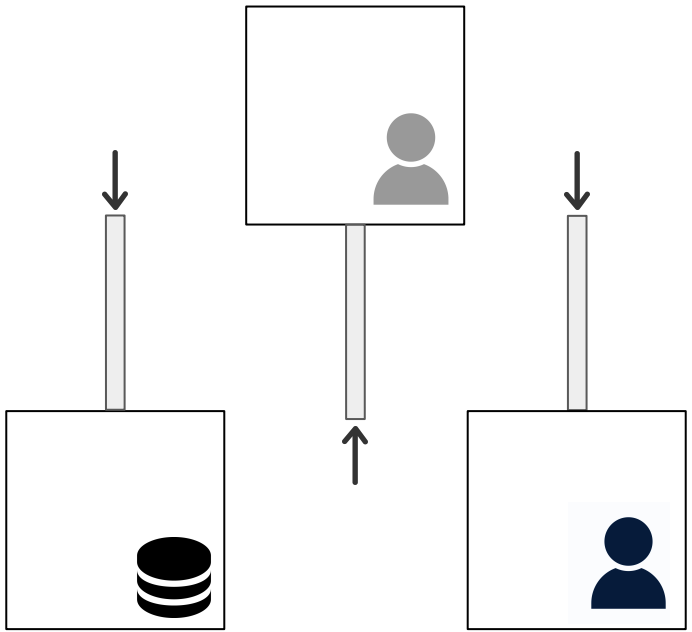


Fig. 1. Map of the Asynchronous Communication Models. A black arrow means "refines". The two axis of refinement are the level of abstraction (data refinement) and the strength of the ordering policy (reduction of non-determinism).

[Chevrou et al. 19]

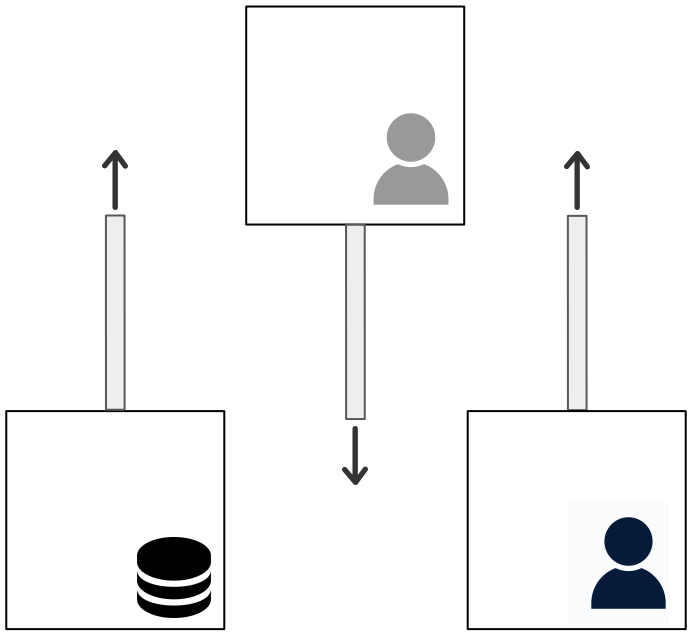
Asynchronous communication



Mailbox (FIFO n-1):

- Actor model: Elixir, Erlang
- Synchronizability and realizability [Etienne and Lozes 17, Muscholl and Delpy 24]

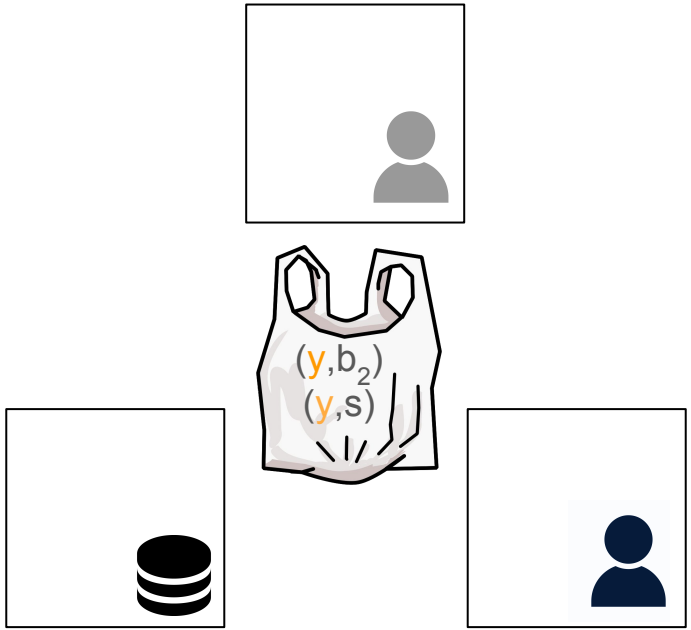
Asynchronous communication



Senderbox (FIFO 1-n):

- Work-stealing in parallel programming

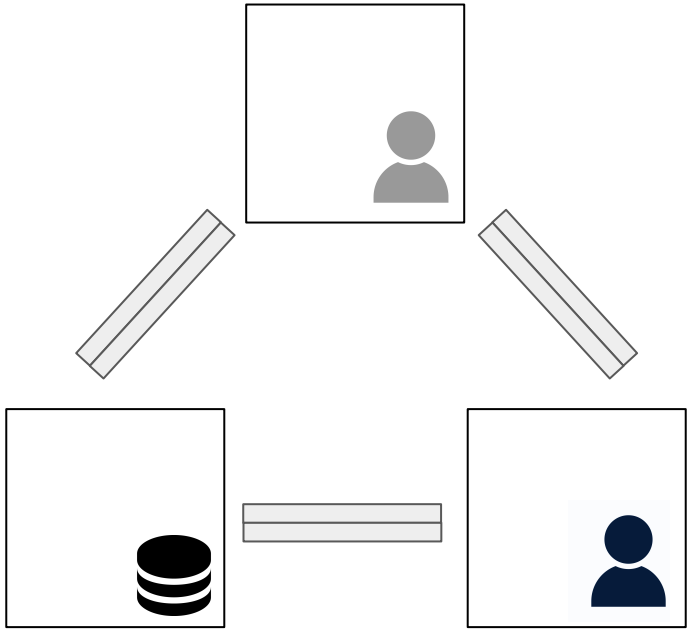
Asynchronous communication



Monobag (async):

- Consensus protocols

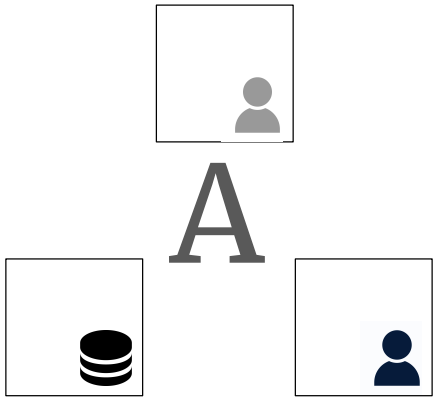
Asynchronous communication



Peer-to-peer FIFO (FIFO 1-1):

- Communicating state machines [Brand and Zafiropulo 83]
- Ubiquitous in theory and verification

What is a network?



A network architecture A is:

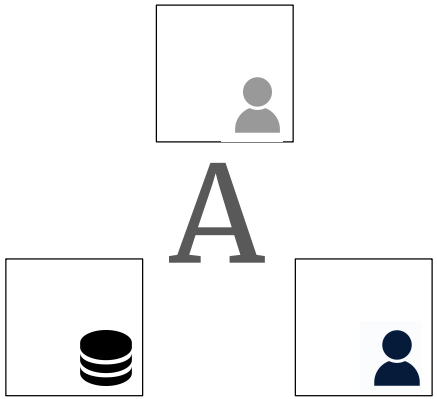
communication topology
(mapping sender and receiver pairs to buffers)

$\times$

buffer data structure
(empty, insert, remove)

What is a network?

Communicating Labeled Transition System (CLTS) = local implementations (controllable) + network architecture $\mathbb{A}$ (non-controllable)



A network architecture $\mathbb{A}$ is:

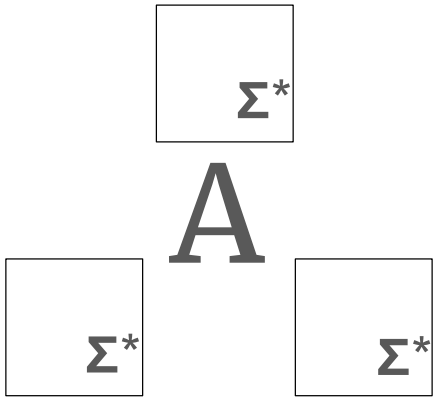
communication topology
(mapping sender and receiver pairs to buffers)

×

buffer data structure
(empty, insert, remove)

What is a network?

Communicating Labeled Transition System (CLTS) = local implementations (controllable) + network architecture (non-controllable) $\mathbb{A}$



A network architecture $\mathbb{A}$ is:

communication topology
(mapping sender and receiver pairs to buffers)

×

buffer data structure
(empty, insert, remove)

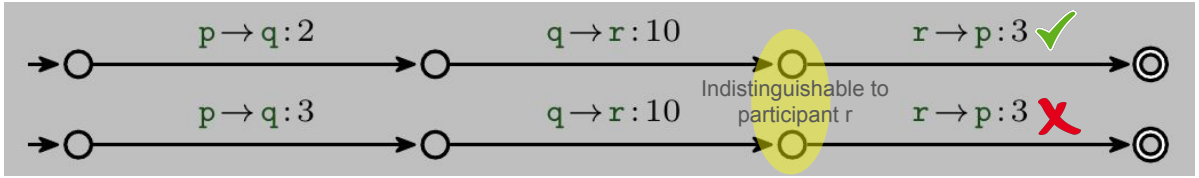
A network is the set of finite and infinite words executable by the *universal* CLTS under network architecture A.

Network-parametric goal

A single **implementability characterization**
and proof of preciseness
for all asynchronous networks

(L(A))

Coherence Conditions under p2p FIFO [Li et al. 25]



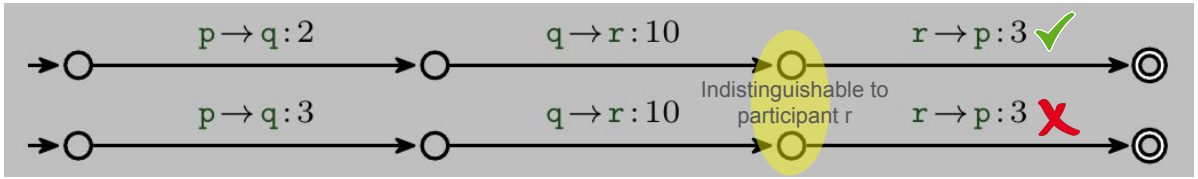
From two locally indistinguishable global states, a participant can either:

- Send a message permissible from both states (Send Coherence)
- Receive a message that distinguishes the two states (Receive Coherence)

but cannot choose between sending or receiving a message (No Mixed Choice)

Theorem. A concrete protocol is implementable if and only if it satisfies the Coherence Conditions.

Generalized Coherence Conditions under $\mathbf{A}$



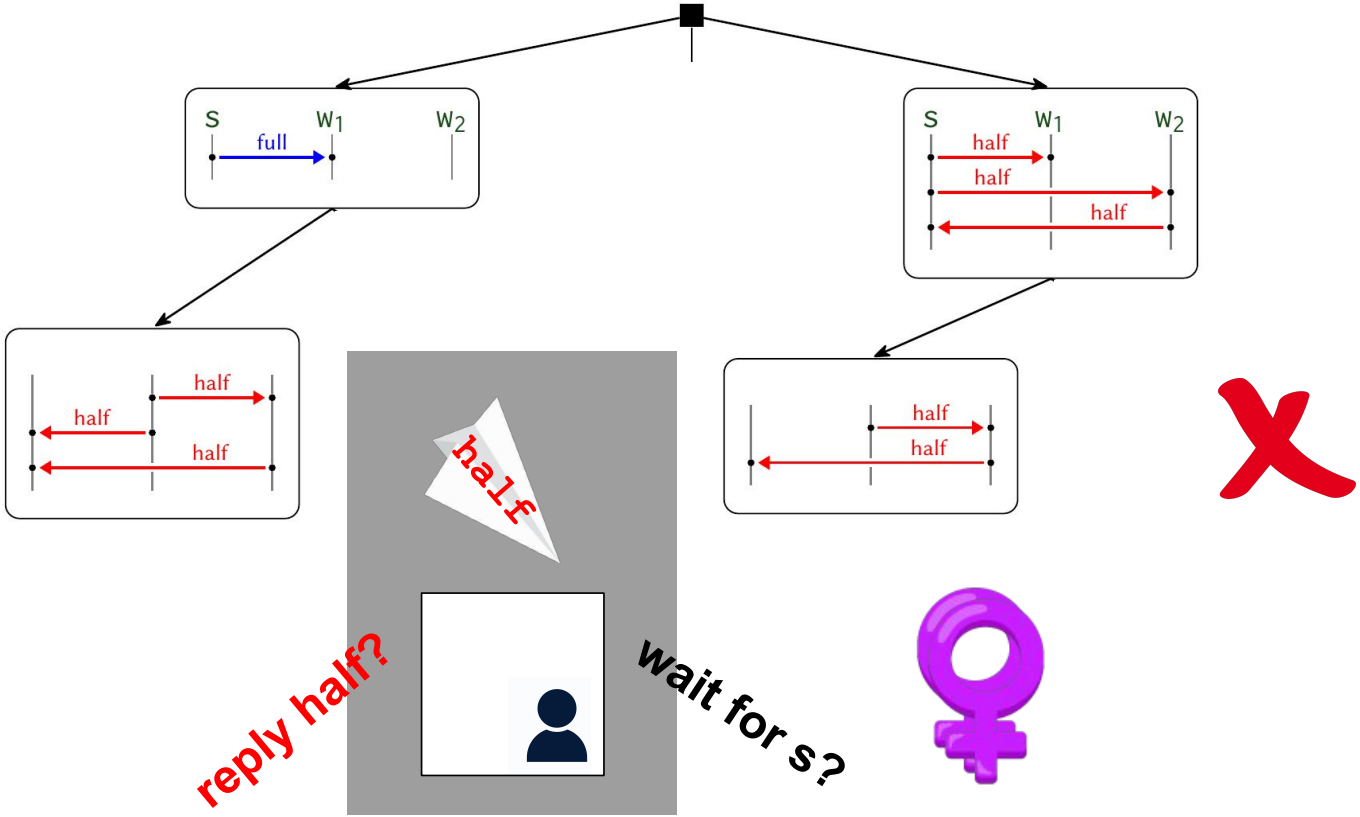
From two locally indistinguishable global states, a participant can either:

- Send a message permissible from both states (Send Coherence)
- **Receive a message under $\mathbf{A}$ that distinguishes the two states (Generalized Receive Coherence)**

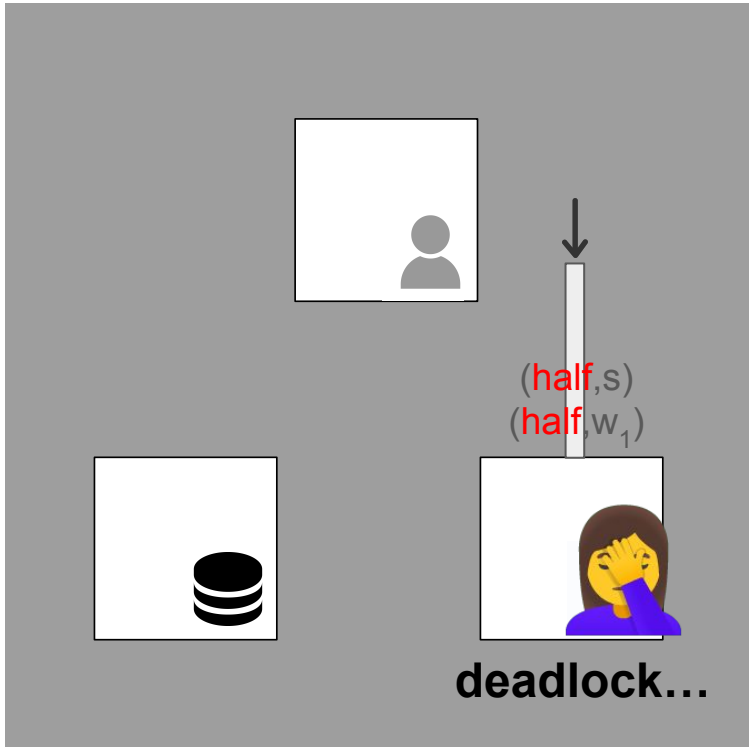
but cannot choose between sending or receiving a message (No Mixed Choice)

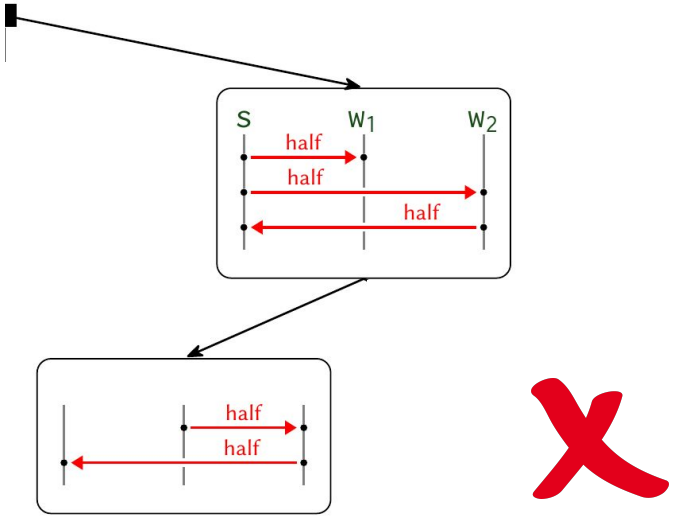
Moreover, the protocol must not contain runs that deadlock under $\mathbf{A}$ despite perfect information (Prefix Extensibility)

Violation of Generalized Receive Coherence (all)



Violation of Prefix Extensibility (mailbox, monobox)





$u_3 = s \triangleright w_1!\text{half} \cdot w_1 \triangleleft s?\text{half} \cdot w_1 \triangleright w_2!\text{half} \cdot s \triangleright w_2!\text{half}$

Network-parametric goal

(Generalized Coherence Conditions under A)

A <u>single</u> implementability characterization
and proof of preciseness
for <u>all</u> **asynchronous networks***

($\mathrm{L}(\mathrm{A})$)

Channel compliance (≈ L(A)) facts

F1 For all $x \in \Sigma_!$, wx is channel-compliant.
F2 For all $\mathsf{p} \neq \mathsf{q} \in \mathcal{P}$ and $m \in \mathcal{V}$, $|w\Downarrow_{\mathsf{p}\triangleright\mathsf{q}!m}| \geq |w\Downarrow_{\mathsf{q}\triangleleft\mathsf{p}?m}|$.
F3 For all $\rho \in \Gamma^*_{sync}$, $\mathtt{split}(\rho)$ is channel-compliant.
F4 For all $\rho \in \Gamma^*_{sync}$, if $w \equiv_{\mathcal{P}} \mathtt{split}(\rho)$, then w is channel-matched.
F5 For all $x \in \Sigma_!$, $y \in \Sigma_?$, if wy is channel-compliant then wxy is channel-compliant.
F6 Let $\alpha, \beta \in \Gamma^*_{sync}$, $\mathsf{p} \neq \mathsf{q} \in \mathcal{P}$ and $m \in \mathcal{V}$ such that for all $\mathsf{p} \in \mathcal{P}$, $w\Downarrow_{\Sigma_\mathsf{p}} \leq \mathtt{split}(\alpha\beta)\Downarrow_{\Sigma_\mathsf{p}}$, and $w\Downarrow_{\Sigma_\mathsf{q}} = \mathtt{split}(\alpha)\Downarrow_{\Sigma_\mathsf{q}}$, and $w \cdot \mathsf{q} \triangleleft \mathsf{p}?m$ is channel-compliant. Then, there exists w' such that w' is compliant with β, $w'\Downarrow_{\Sigma_\mathsf{q}} = \varepsilon$ and $w' \cdot \mathsf{q} \triangleleft \mathsf{p}?m$ is channel-compliant.

Network-parametric goal

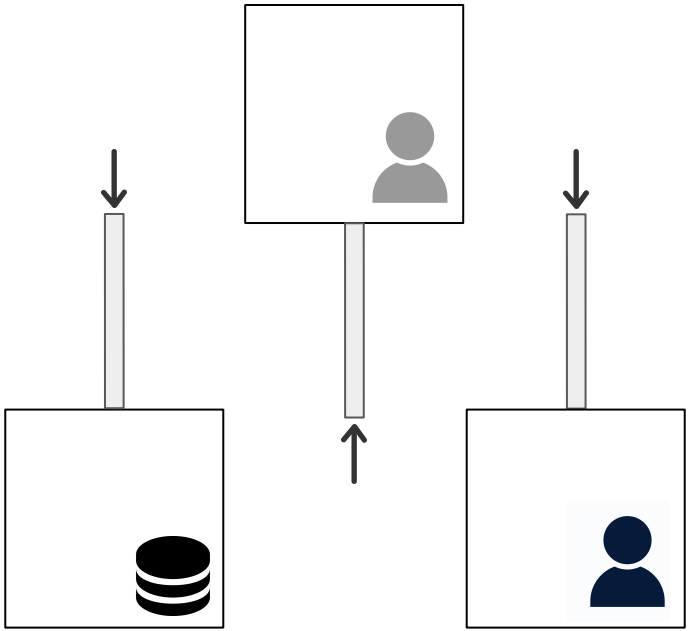
(Generalized Coherence Conditions under $\mathbb{A}$)

A single implementability characterization
and **proof of preciseness**
for all asynchronous networks*

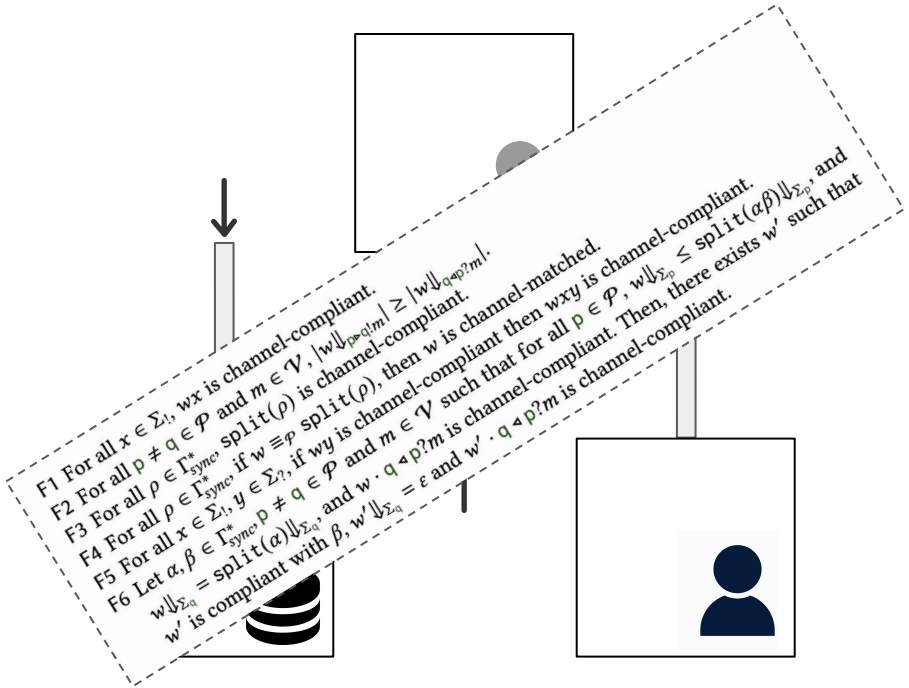
(network architectures $\mathbb{A}$ satisfying F1-F6)

Theorem. Let $\mathbb{A}$ satisfy F1-F6. Then, G is implementable under $\mathbb{A}$ iff G satisfies Generalized Coherence Conditions under $\mathbb{A}$.

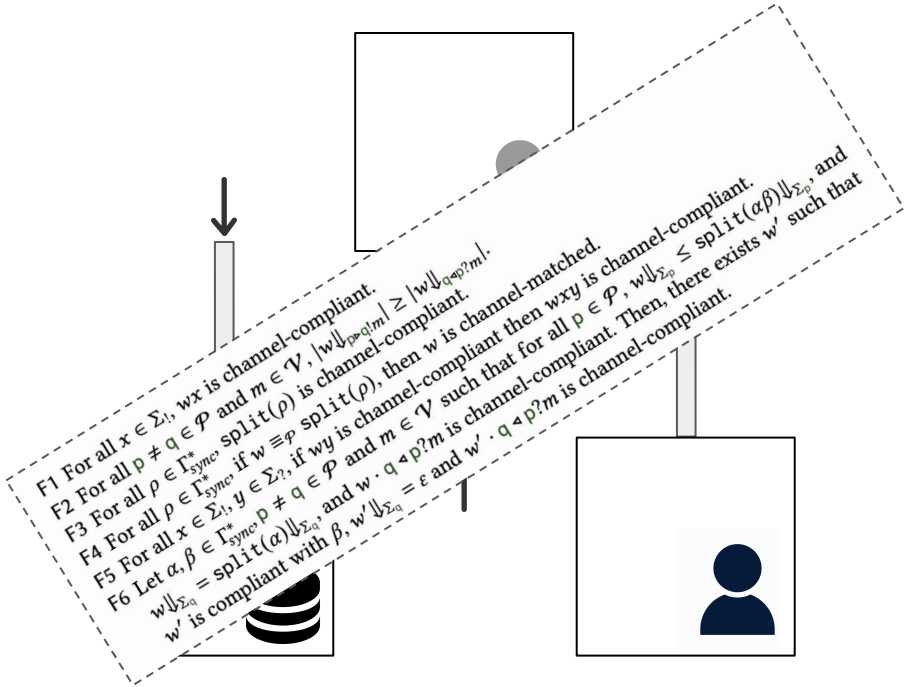
Using the framework

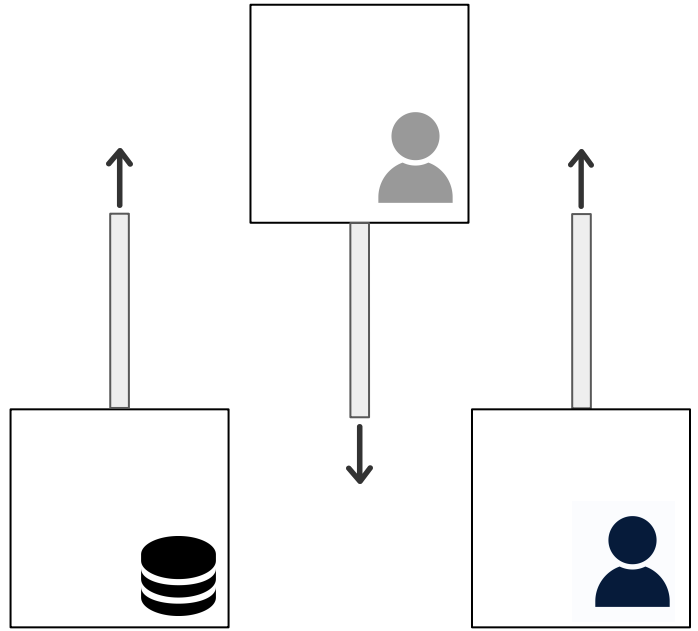


Using the framework



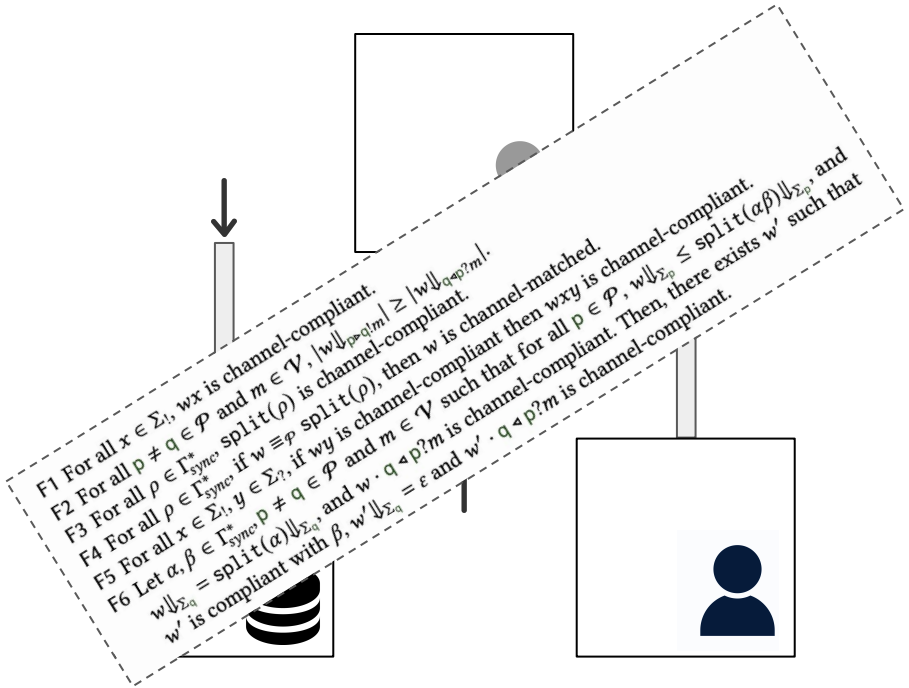
Using the framework

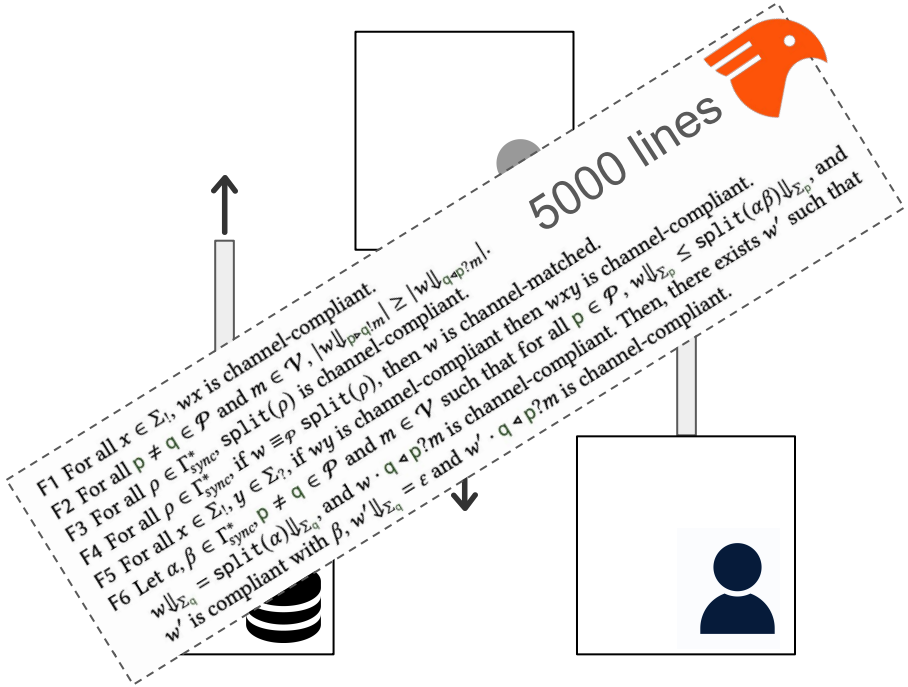




L(mailbox) ≠ L(senderbox)

Using the framework





L(mailbox) ≠ L(senderbox)

Buffer axioms

B1 $b \bullet ins(x)$ is defined.
B2 $b_0 \bullet ins(x) \bullet rem(x)$ is defined.
B3 $b_0 \bullet rem(x)$ is undefined.
B4 If $b \bullet rem(x)$ is defined, then $b \bullet ins(y) \bullet rem(x)$ is defined.
B5 If $b \bullet ins(x) \bullet \delta$ is defined and $rem(x)$ does not occur in δ, then $b \bullet \delta$ is defined.
B6 If $b \bullet rem(x)$ is defined, then $b \bullet ins(x) \bullet rem(x) = b \bullet rem(x) \bullet ins(x)$.
B7 If $x \neq y$, then $b \bullet ins(x) \bullet rem(y) = b \bullet rem(y) \bullet ins(x)$.
B8 If $b \bullet rem(x)$ is undefined, $b \bullet ins(x) \bullet \delta \bullet rem(x)$ is defined, and $rem(x)$ does not occur in δ, then $b \bullet ins(x) \bullet \delta \bullet rem(x) = b \bullet \delta$.

Buffer axioms

B1 $b \bullet ins(x)$ is defined. ~~bounded channels~~
B2 $b_0 \bullet ins(x) \bullet rem(x)$ is defined.
B3 $b_0 \bullet rem(x)$ is undefined.
B4 If $b \bullet rem(x)$ is defined, then $b \bullet ins(y) \bullet rem(x)$ is defined.
B5 If $b \bullet ins(x) \bullet \delta$ is defined and $rem(x)$ does not occur in δ, then $b \bullet \delta$ is defined.
B6 If $b \bullet rem(x)$ is defined, then $b \bullet ins(x) \bullet rem(x) = b \bullet rem(x) \bullet ins(x)$.
B7 If $x \neq y$, then $b \bullet ins(x) \bullet rem(y) = b \bullet rem(y) \bullet ins(x)$.
B8 If $b \bullet rem(x)$ is undefined, $b \bullet ins(x) \bullet \delta \bullet rem(x)$ is defined, and $rem(x)$ does not occur in δ, then $b \bullet ins(x) \bullet \delta \bullet rem(x) = b \bullet \delta$.

Buffer axioms

B1 $b \bullet ins(x)$ is defined. ~~bounded channels~~

B2 $b_0 \bullet ins(x) \bullet rem(x)$ is defined.

B3 $b_0 \bullet rem(x)$ is undefined.

B4 If $b \bullet rem(x)$ is defined, then $b \bullet ins(y) \bullet rem(x)$ is defined. ~~priority queues, stacks~~

B5 If $b \bullet ins(x) \bullet \delta$ is defined and $rem(x)$ does not occur in δ, then $b \bullet \delta$ is defined.

B6 If $b \bullet rem(x)$ is defined, then $b \bullet ins(x) \bullet rem(x) = b \bullet rem(x) \bullet ins(x)$.

B7 If $x \neq y$, then $b \bullet ins(x) \bullet rem(y) = b \bullet rem(y) \bullet ins(x)$.

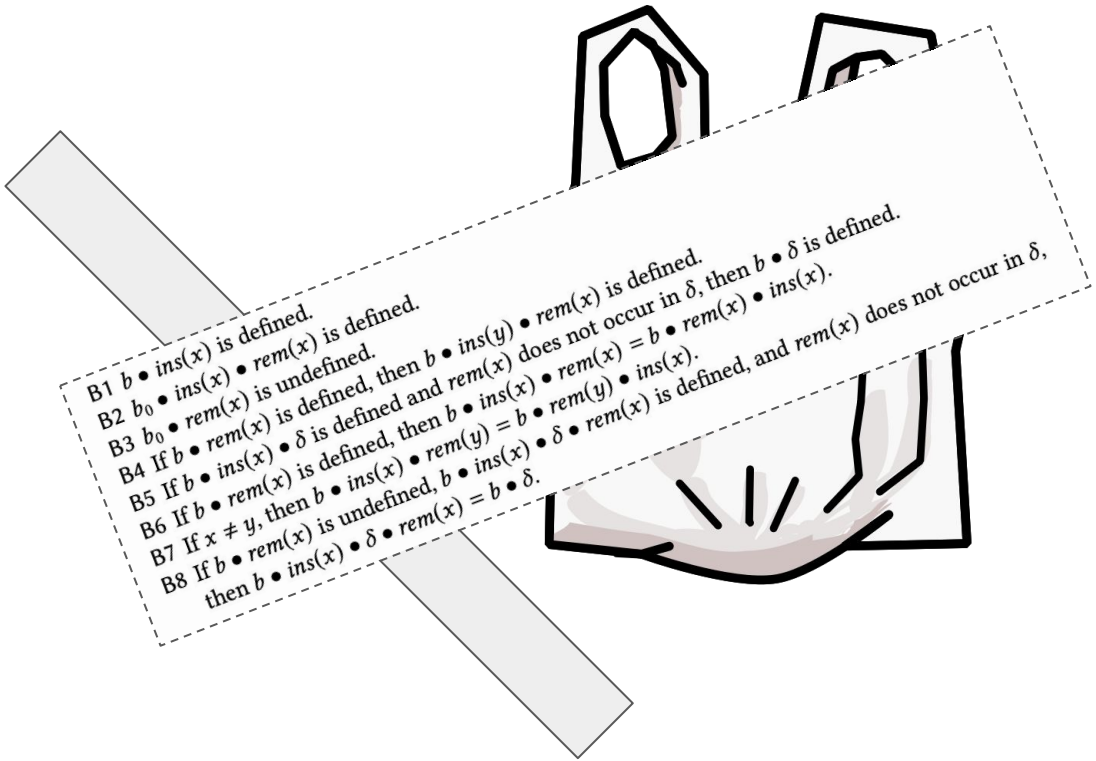
B8 If $b \bullet rem(x)$ is undefined, $b \bullet ins(x) \bullet \delta \bullet rem(x)$ is defined, and $rem(x)$ does not occur in δ, then $b \bullet ins(x) \bullet \delta \bullet rem(x) = b \bullet \delta$.

Buffer axioms

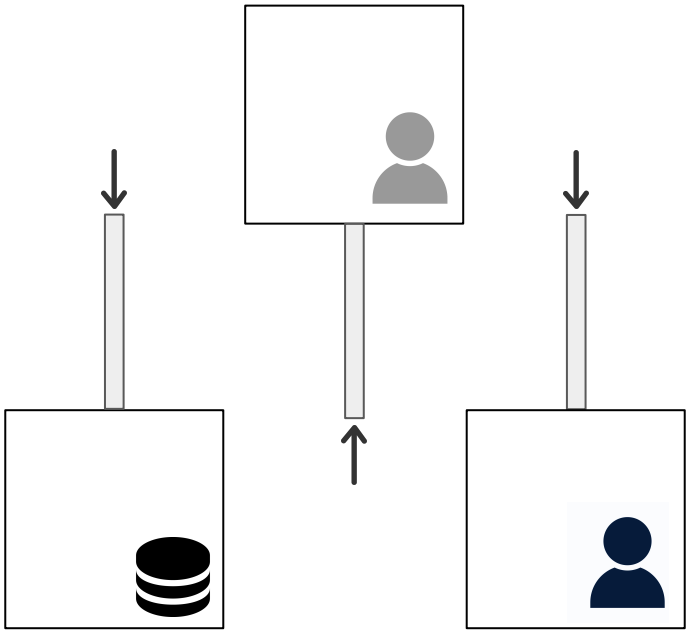
B1 $b \bullet ins(x)$ is defined. ~~bounded channels~~
B2 $b_0 \bullet ins(x) \bullet rem(x)$ is defined.
B3 $b_0 \bullet rem(x)$ is undefined.
B4 If $b \bullet rem(x)$ is defined, then $b \bullet ins(y) \bullet rem(x)$ is defined. ~~priority queues, stacks~~
B5 If $b \bullet ins(x) \bullet \delta$ is defined and $rem(x)$ does not occur in δ, then $b \bullet \delta$ is defined.
B6 If $b \bullet rem(x)$ is defined, then $b \bullet ins(x) \bullet rem(x) = b \bullet rem(x) \bullet ins(x)$.
B7 If $x \neq y$, then $b \bullet ins(x) \bullet rem(y) = b \bullet rem(y) \bullet ins(x)$.
B8 If $b \bullet rem(x)$ is undefined, $b \bullet ins(x) \bullet \delta \bullet rem(x)$ is defined, and $rem(x)$ does not occur in δ, then $b \bullet ins(x) \bullet \delta \bullet rem(x) = b \bullet \delta$.

FIFO queues, multisets

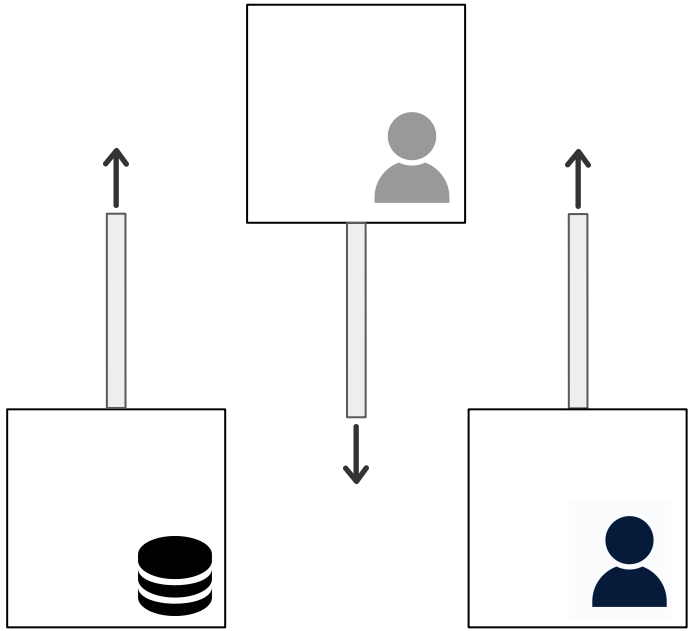
Using the framework, take two



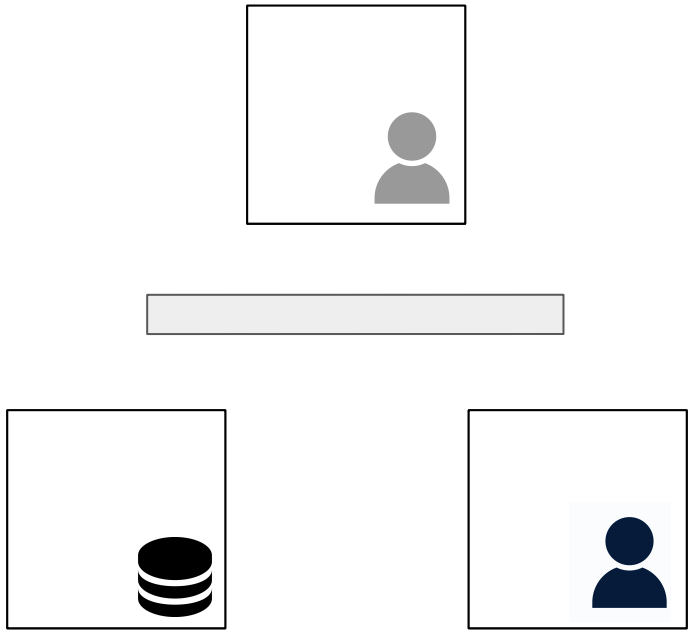
Using the framework, take two



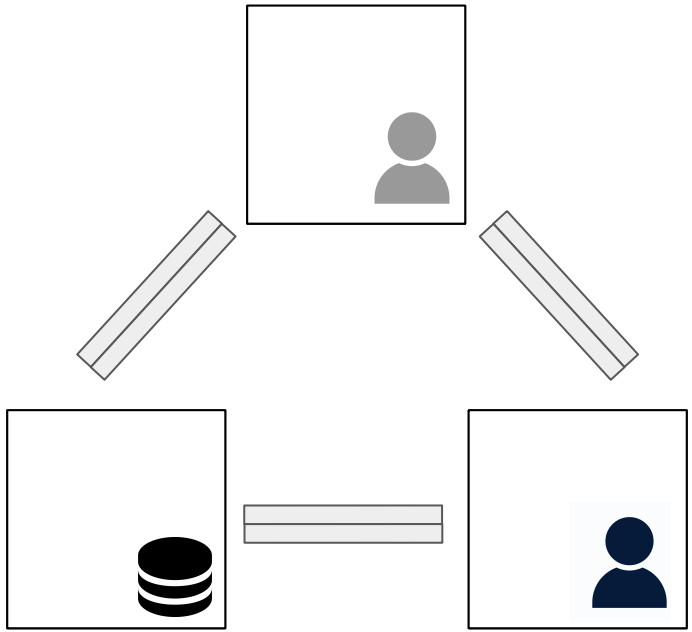
Using the framework, take two



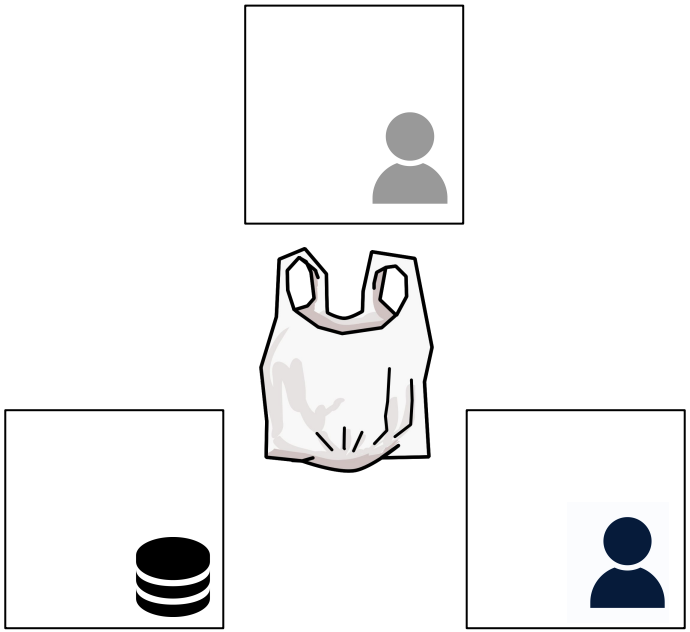
Using the framework, take two



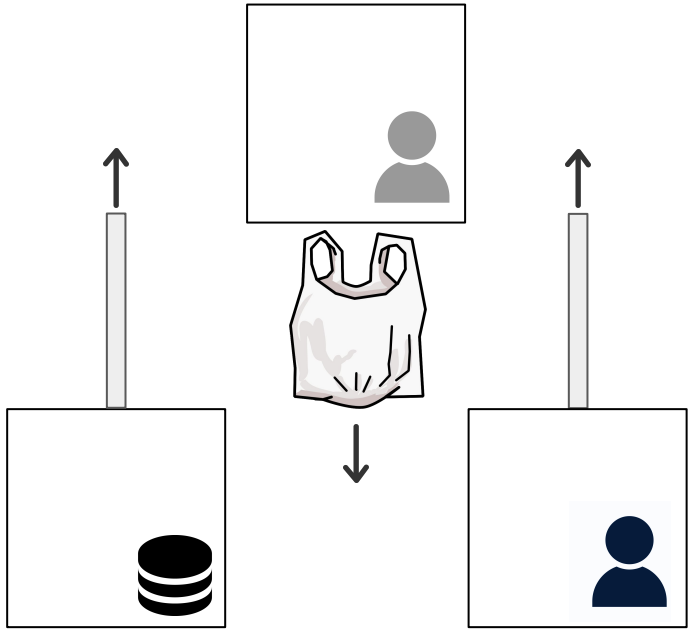
Using the framework, take two



Using the framework, take two



Using the framework, take two



SPROUT(A) [Li et al. CAV'25]

```
Initial state: (0)
Initial register assignments: ry=0, rc=0, rz1=0, rz2=0
(0) B1->S:y{(y>987000000000/\y<9880000000000)/\ry'=y} (1)
(1) B1->B2:y{y=ry} (2)
(2) S->B1:z{z>0/\rc'=z} (3)
(3) B1->B2:b1{b1>rz1/\rz1'=b1} (4)
(4) B2->S:quit{quit=0} (5)
(5) S->B1:quit{quit=0} (6)
(4) B2->B1:b2{b2>rz2/\b2<rc/\rz2'=b2} (7)
(7) B1->S:succ{succ=1/\rz1+rz2>=rc} (8)
(8) S->B2:succ{succ=1} (6)
(7) B1->B2:cont{cont=2/\rz1+rz2<rc} (3)
Final states: (6)
```

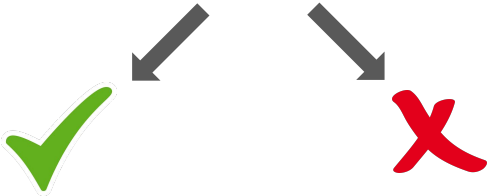
+ $A \in$ {p2pbox, senderbox, mailbox, monobox, bag}

sound and complete μCLP reduction (first order fixpoint logic)

Generalized Symbolic Coherence Conditions

calls solver

MuVal [Unno et al. 23]



Efficiency

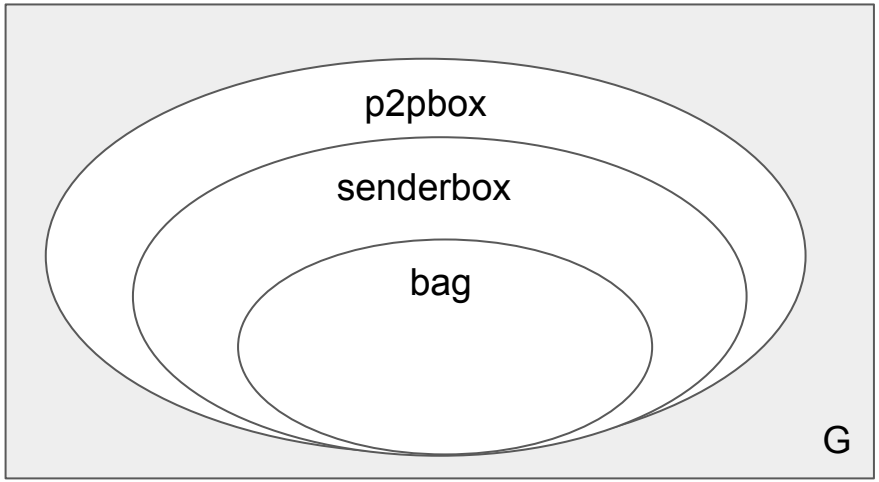
[Li et al. CAV'25]

Source	Example	$\|\mathcal{P}\|$	p2p	Time	sb	Time	mb	Time	monob	Time	bag	Time
[87]	Calculator	2	✓	0.6s	✓	0.7s	✓	0.7s	✓	0.8s	×	16.5s
	Fibonacci	2	✓	0.5s	✓	0.5s	✓	0.5s	✓	0.5s	×	3.3s
	HigherLower	3	✓	15.8s	✓	13.6s	×	17.0s	×	18.5s	×	45.6s
	HTTP	2	✓	0.5s	✓	0.4s	✓	0.4s	✓	0.5s	×	24.4s
	Negotiation	2	✓	1.0s	✓	1.0s	✓	1.0s	✓	1.0s	×	26.1s
	OnlineWallet	3	✓	17.5s	✓	16.6s	✓	16.5s	✓	20.6s	×	100.0s
	SH	3	✓	237.1s	✓	244.0s	✓	256.1s	✓	257.2s	?	T/O
	Ticket	2	✓	0.6s	✓	0.6s	✓	0.6s	✓	0.6s	×	3.8s
	TravelAgency	2	✓	9.6s	✓	9.7s	×	12.2s	×	11.6s	×	18.3s
	TwoBuyer	3	✓	4.2s	✓	4.0s	×	6.3s	×	6.3s	✓	9.3s
[83]	DoubleBuffering	3	✓	1.5s	✓	1.5s	×	2.4s	×	2.3s	✓	2.5s
	OAuth	3	✓	6.5s	✓	6.6s	✓	10.5s	✓	10.3s	×	18.1s
	PlusMinus	3	✓	5.5s	✓	5.3s	×	7.7s	×	8.6s	✓	13.2s
	RingMax	7	✓	3.7s	✓	3.6s	✓	5.3s	✓	6.8s	✓	16.5s
	SimpleAuth	2	✓	0.5s	✓	0.5s	✓	0.5s	✓	0.5s	×	4.7s
	TravelAgency2	2	✓	0.5s	✓	0.5s	✓	0.5s	✓	0.5s	✓	3.7s
[56]	send-validity-yes	4	✓	1.9s	✓	2.7s	✓	2.7s	✓	2.7s	✓	3.6s
	send-validity-no	4	×	1.9s	×	1.9s	×	2.7s	×	2.7s	×	3.6s
	receive-validity-yes	3	✓	5.3s	✓	5.3s	✓	6.5s	✓	5.9s	✓	7.8s
	receive-validity-no	3	×	3.7s	×	3.8s	×	4.6s	×	4.0s	×	5.4s
[57]	symbolic-two-bidder-yes	3	✓	24.0s	✓	21.6s	✓	25.2s	✓	22.0s	✓	31.7s
	symbolic-two-bidder-no1	3	×	23.9s	×	20.3s	×	22.9s	×	18.3s	×	29.5s
	figure12-yes	3	✓	2.0s	✓	2.0s	✓	8.7s	✓	8.5s	✓	12.8s
	figure12-no	3	×	3.0s	×	3.0s	×	9.7s	×	10.7s	×	13.8s
	symbolic-send-validity-yes	4	✓	6.6s	✓	6.5s	✓	10.0s	✓	10.5s	✓	16.0s
	symbolic-send-validity-no	4	×	5.6s	×	5.4s	×	8.3s	×	8.8s	×	13.2s

...

Implementability relationships

Source	Example	$\lvert\mathcal{P}\rvert$	p2p	Time	sb	Time	mb	Time	monob	Time	bag	Time
new	bag-no-p2p-yes	2	✓	0.4s	✓	0.4s	✓	3.4s	✓	0.6s	✗	1.8s
	mb-no-p2p-yes	3	✓	0.9s	✓	0.9s	✗	1.3s	✗	1.3s	✓	1.3s
	p2p-no-sb-yes	4	✗	4.0s	✓	4.1s	✗	5.3s	✗	5.3s	✗	6.7s
	mb-no-monob-no	3	✗	3.2s	✓	3.2s	✗	4.1s	✗	4.1s	✗	5.4s
	monob-no-mb-no	4	✓	1.3s	✓	1.3s	✗	1.9s	✗	1.9s	✓	1.9s
	Average			6.7s		5.8s		7.6s		7.1s		12.6s

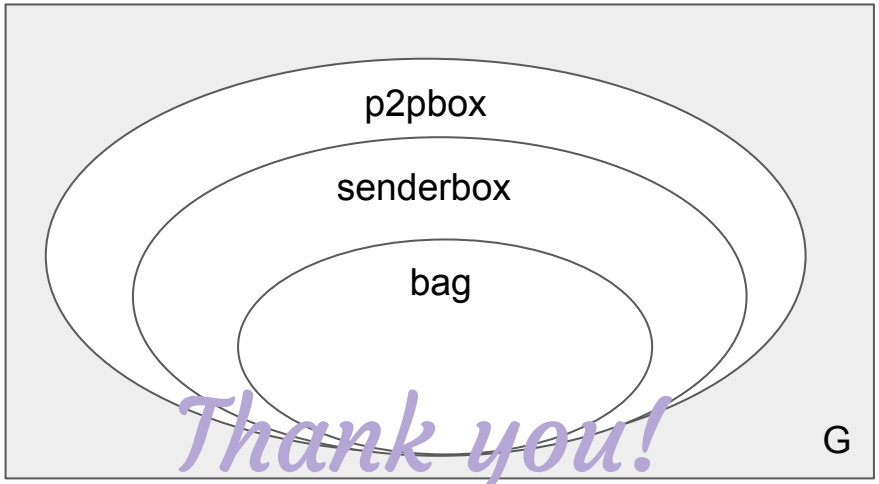


Corollary.

Implementability relationships

Source	Example	$\lvert\mathcal{P}\rvert$	p2p	Time	sb	Time	mb	Time	monob	Time	bag	Time
new	bag-no-p2p-yes	2	✓	0.4s	✓	0.4s	✓	3.4s	✓	0.6s	✗	1.8s
	mb-no-p2p-yes	3	✓	0.9s	✓	0.9s	✗	1.3s	✗	1.3s	✓	1.3s
	p2p-no-sb-yes	4	✗	4.0s	✓	4.1s	✗	5.3s	✗	5.3s	✗	6.7s
	mb-no-monob-no	3	✗	3.2s	✓	3.2s	✗	4.1s	✗	4.1s	✗	5.4s
	monob-no-mb-no	4	✓	1.3s	✓	1.3s	✗	1.9s	✗	1.9s	✓	1.9s
	Average			6.7s		5.8s		7.6s		7.1s		12.6s

Corollary.



Thank you!