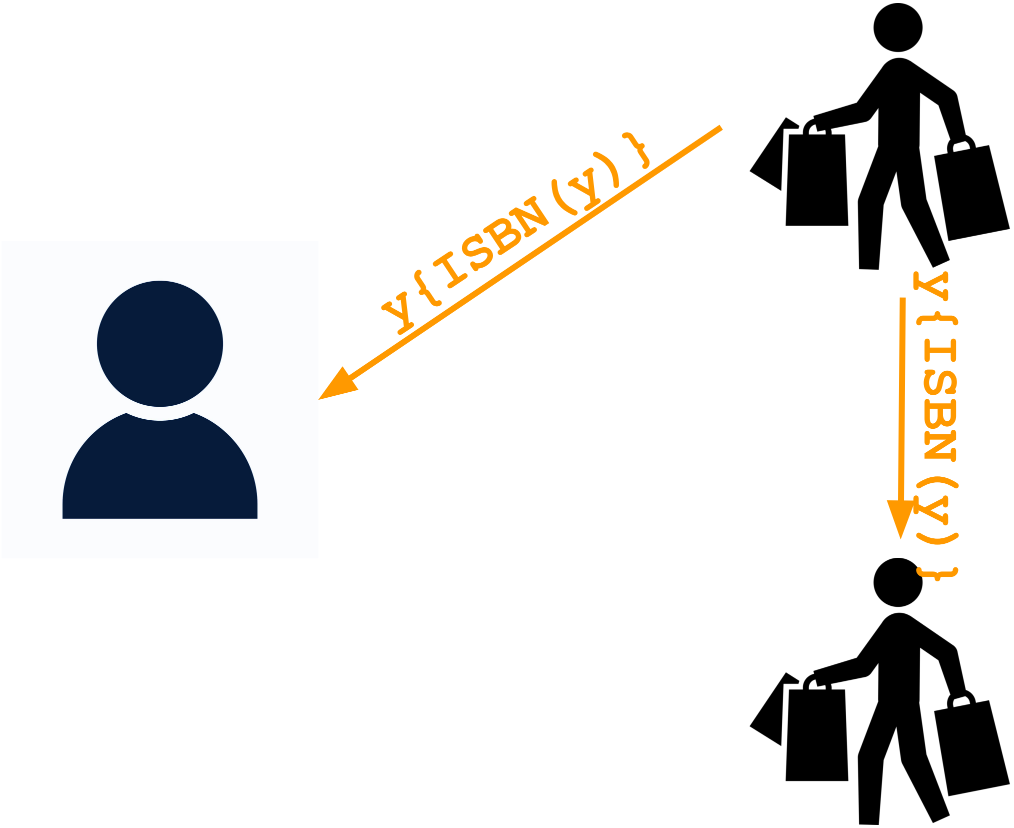
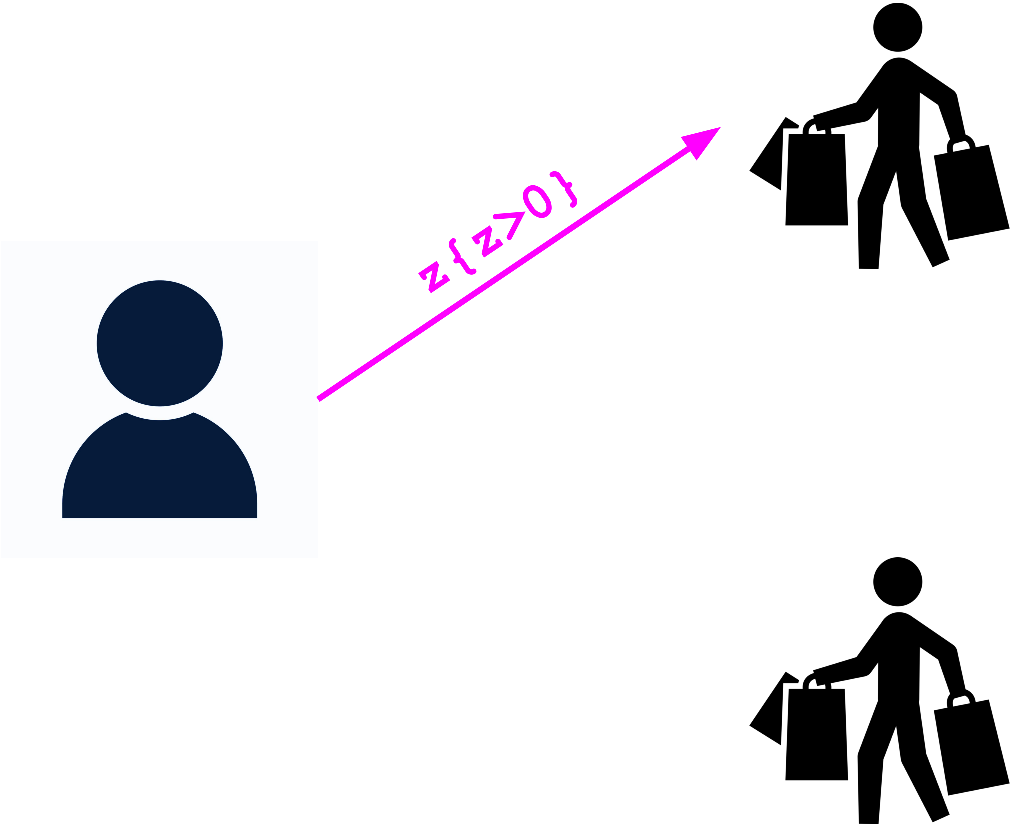
SPROUT:
A Verifier for Symbolic Multiparty Protocols

Elaine Li   Felix Stutz   Thomas Wies   Damien Zufferey

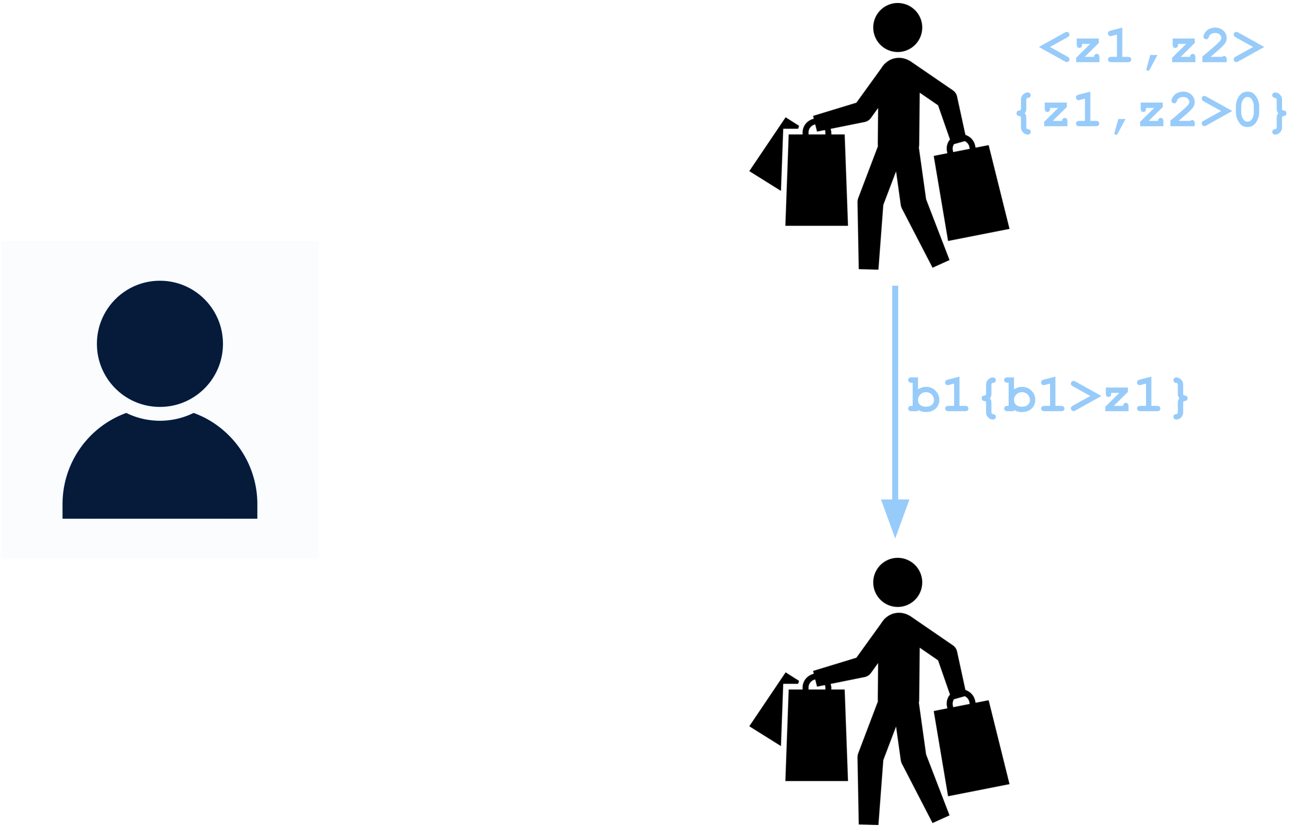
Example: two-bidder protocol



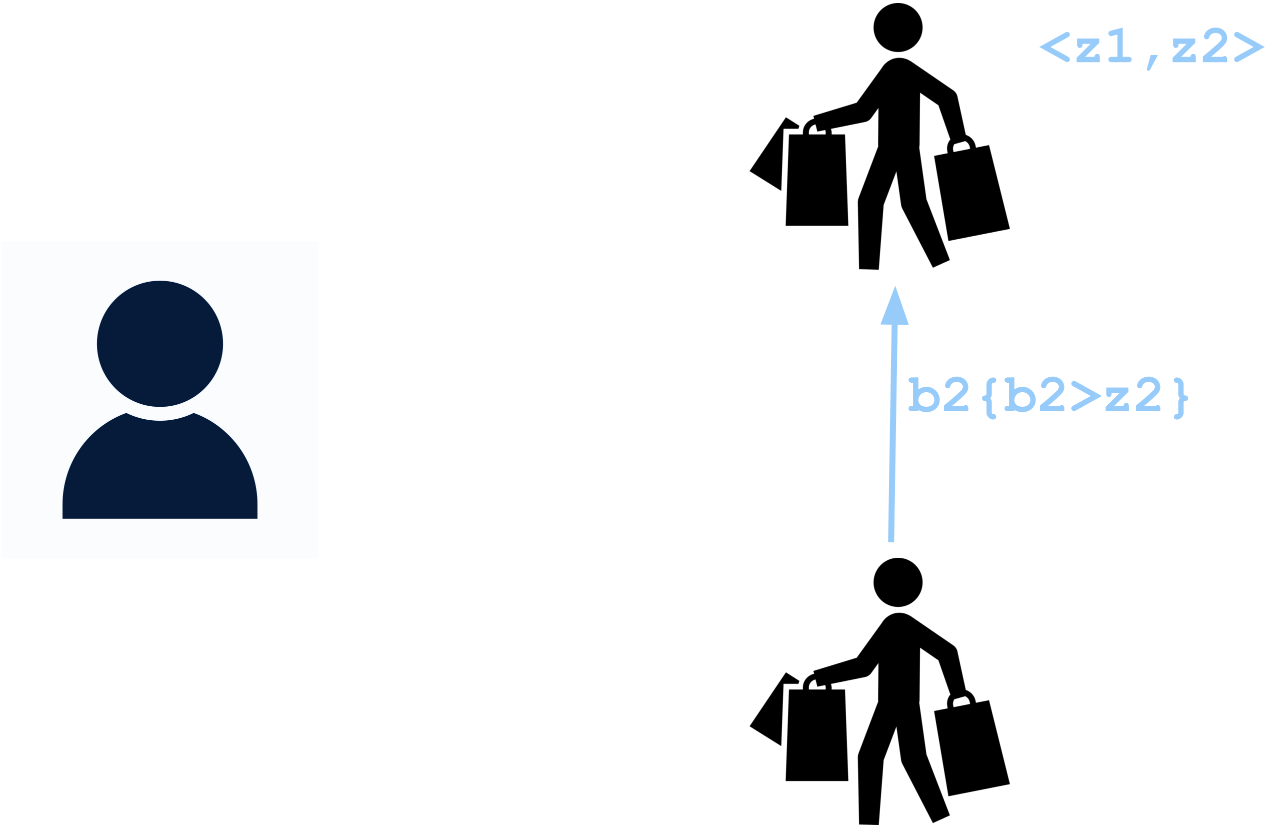
Example: two-bidder protocol



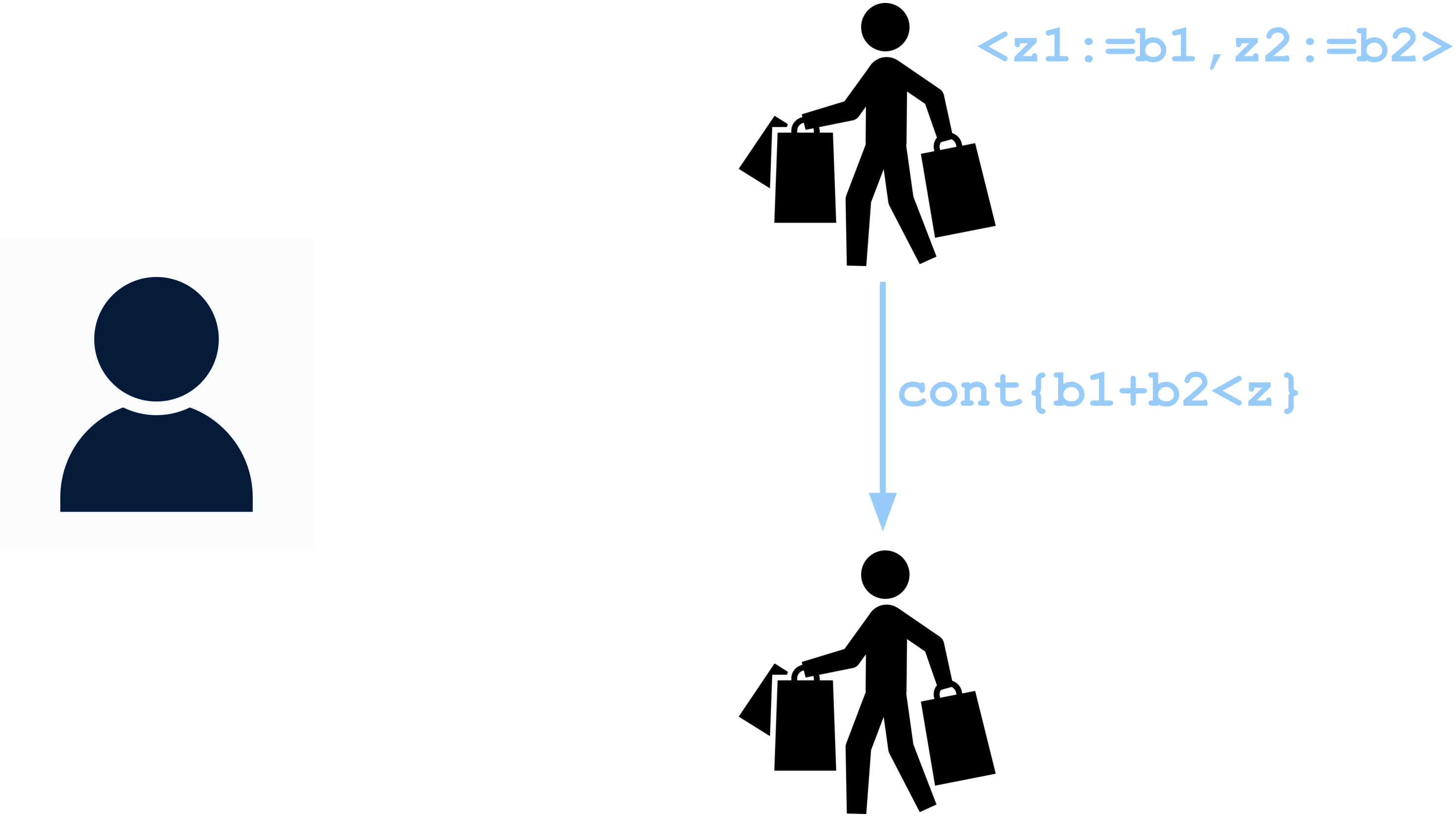
Example: two-bidder protocol



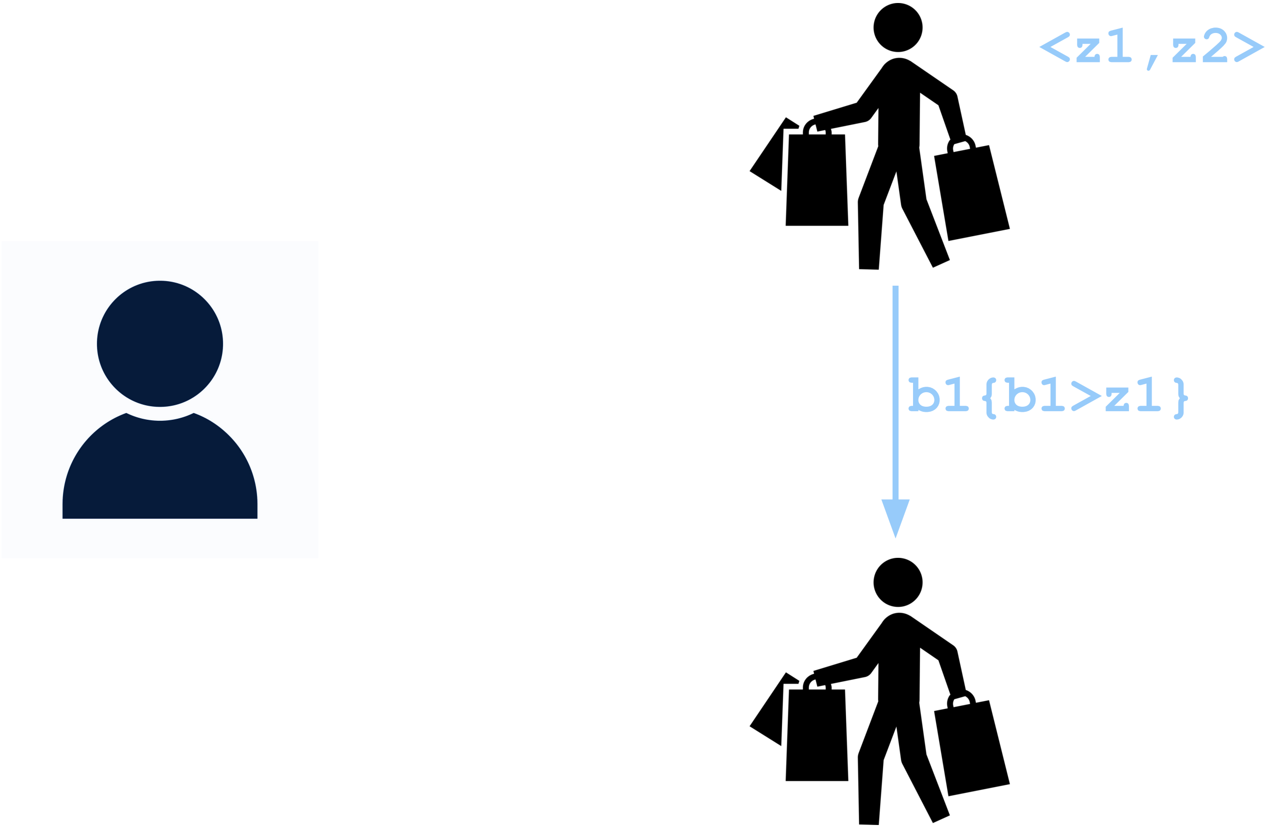
Example: two-bidder protocol



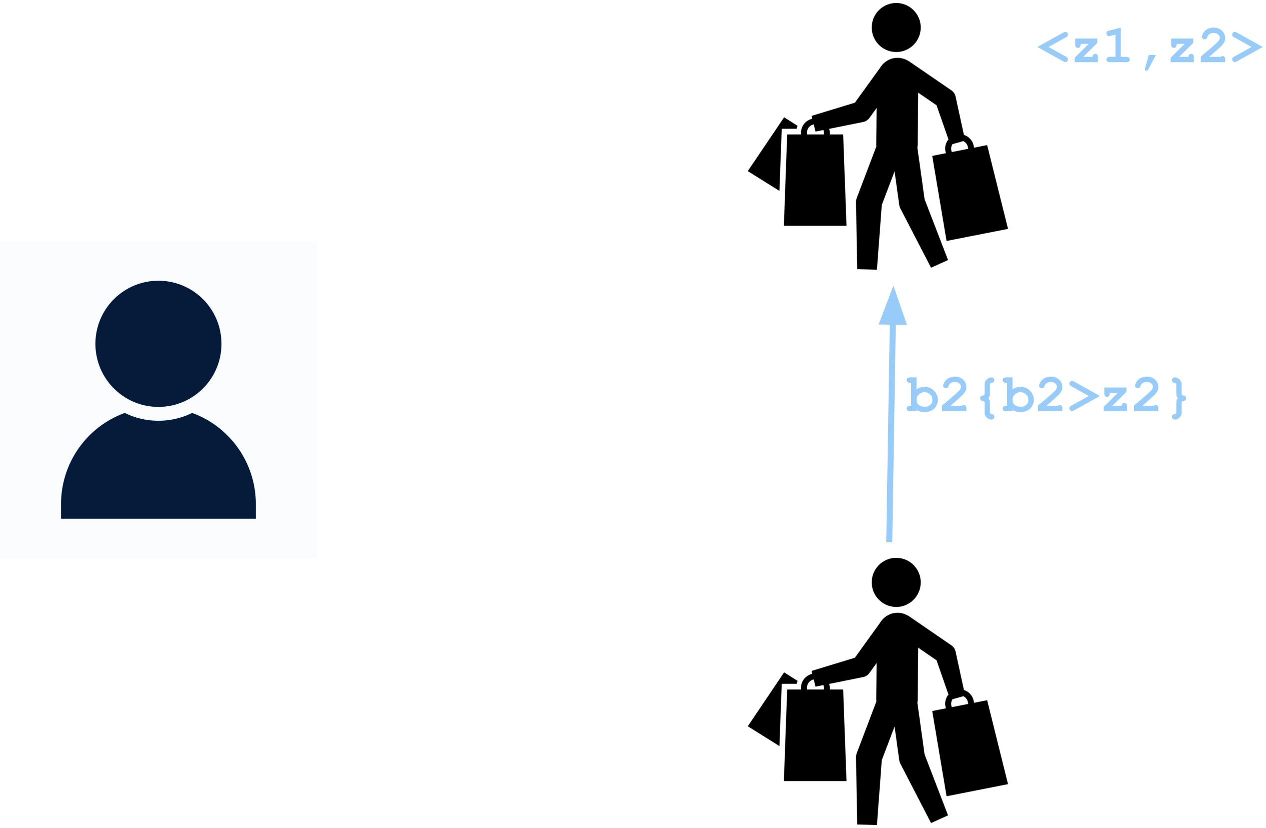
Example: two-bidder protocol



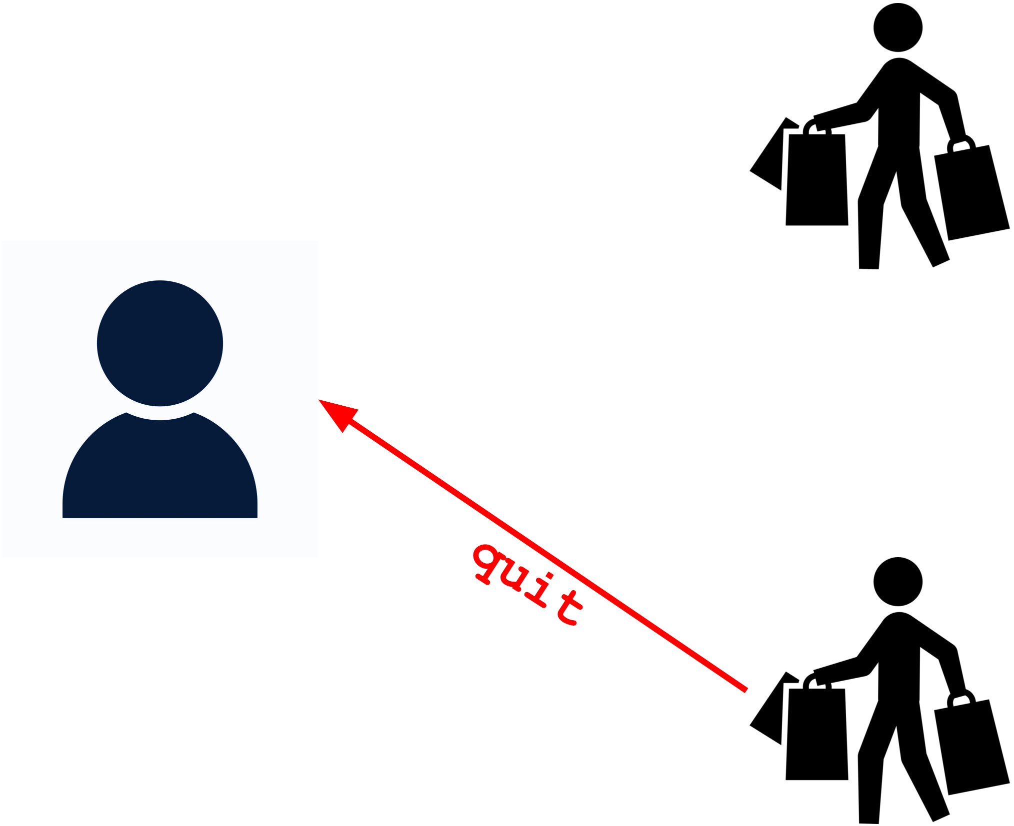
Example: two-bidder protocol



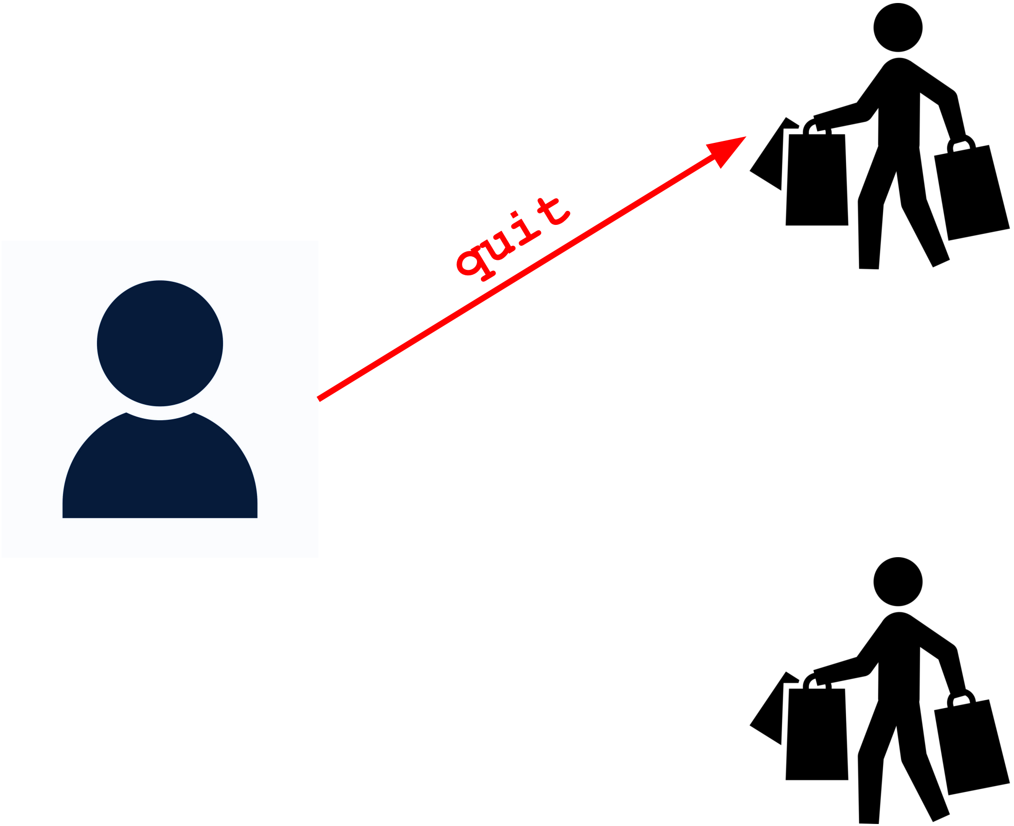
Example: two-bidder protocol



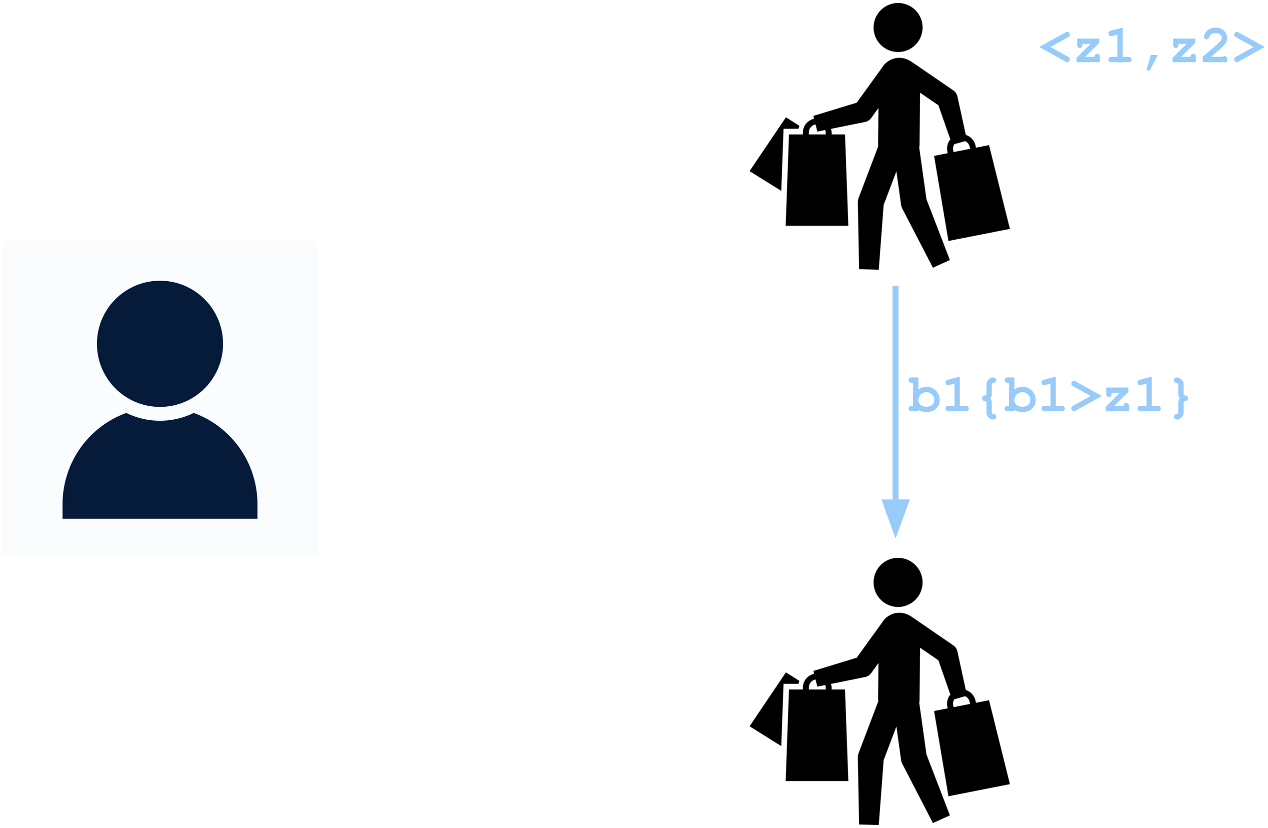
Example: two-bidder protocol



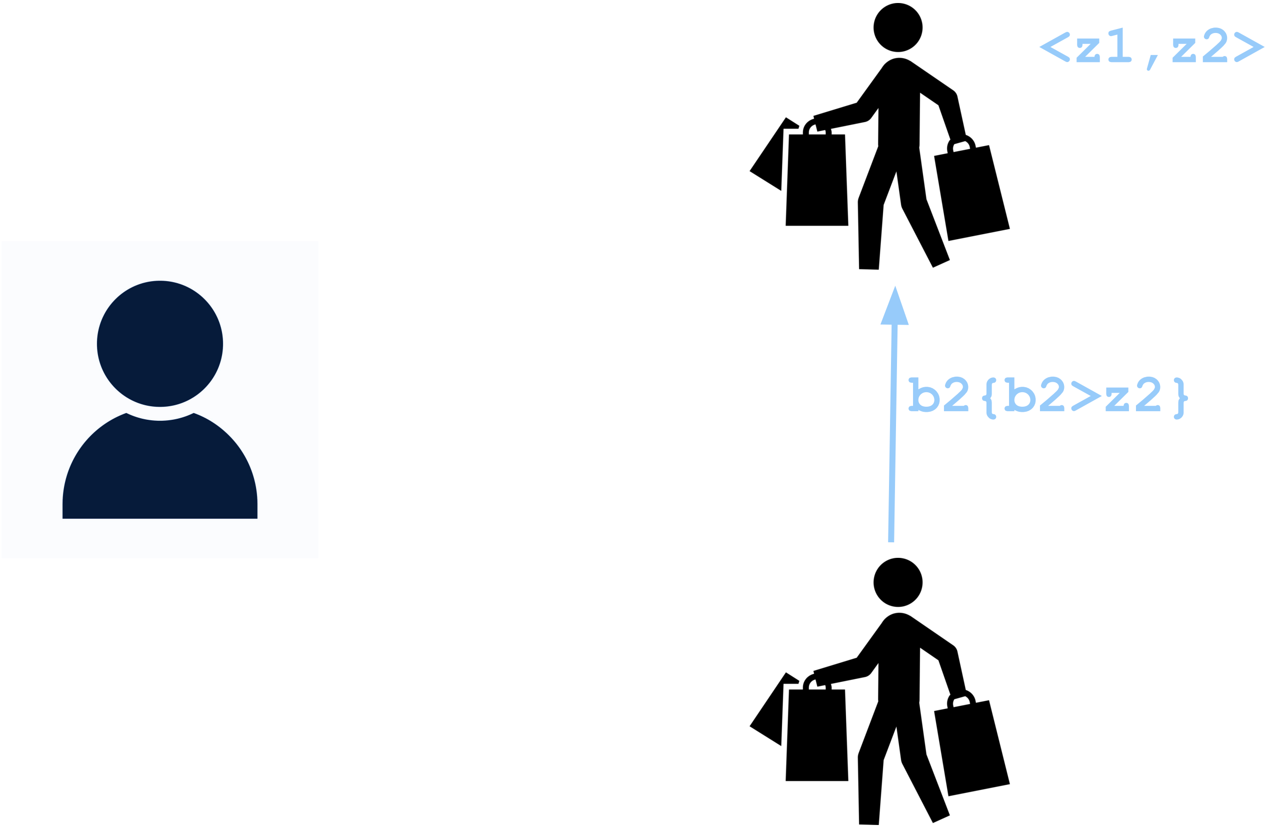
Example: two-bidder protocol



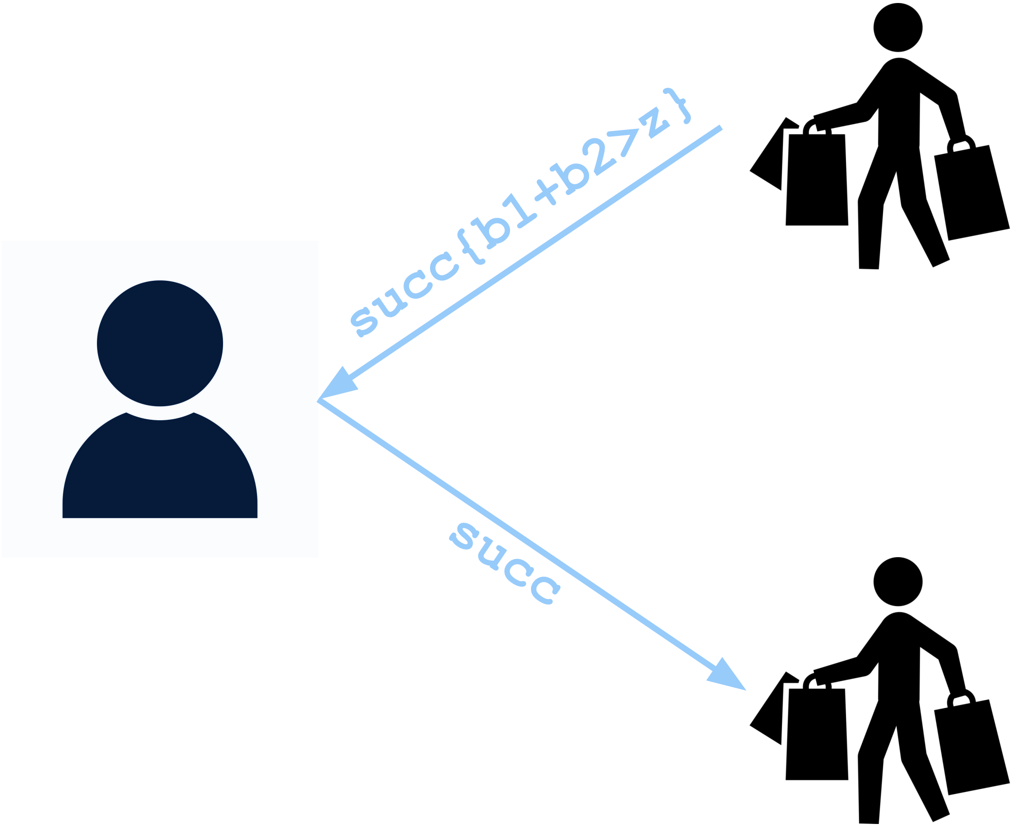
Example: two-bidder protocol



Example: two-bidder protocol



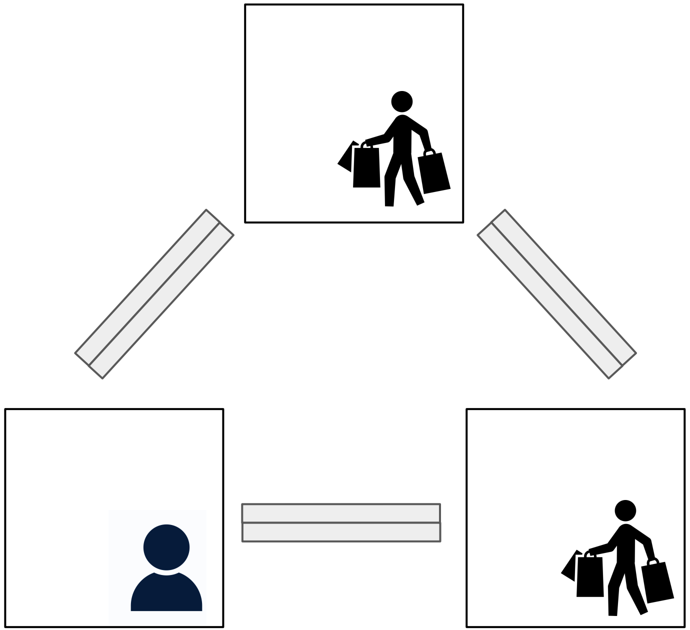
Example: two-bidder protocol



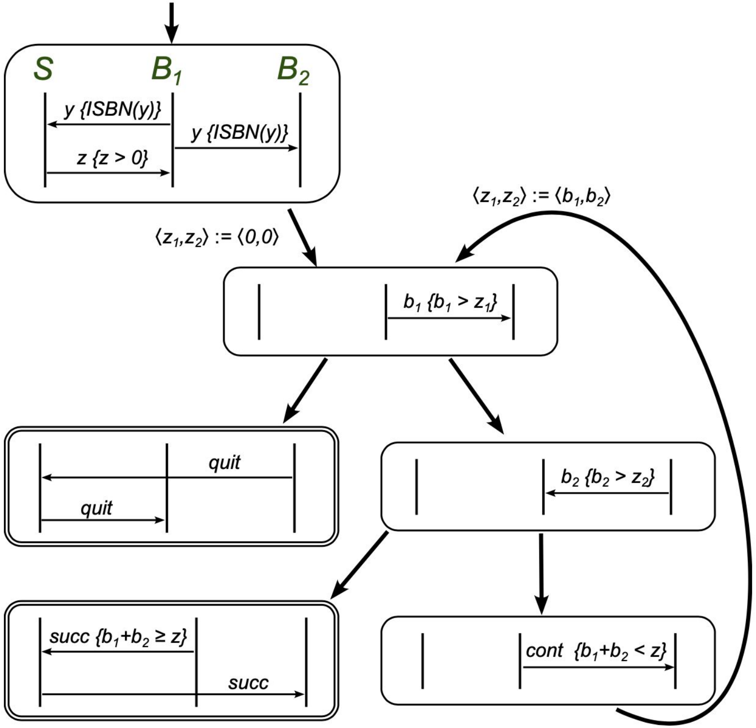
Motivation

Asynchronous, message-passing programs are challenging to implement individually

- Communication errors e.g. orphan messages, unspecified receptions
- Deadlocks



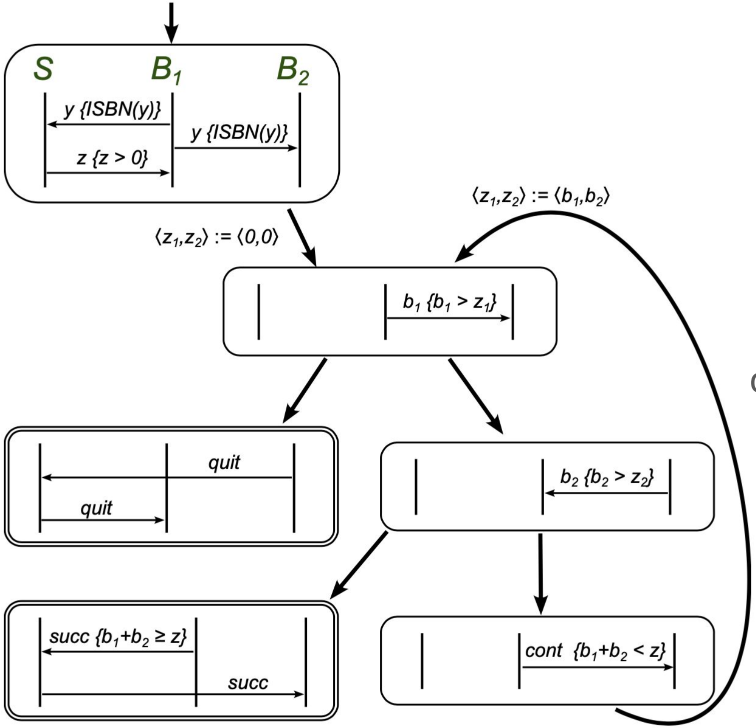
Motivation

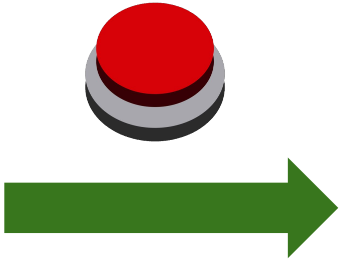


Global protocols are **synchronous** specifications of **all** participants' behaviors

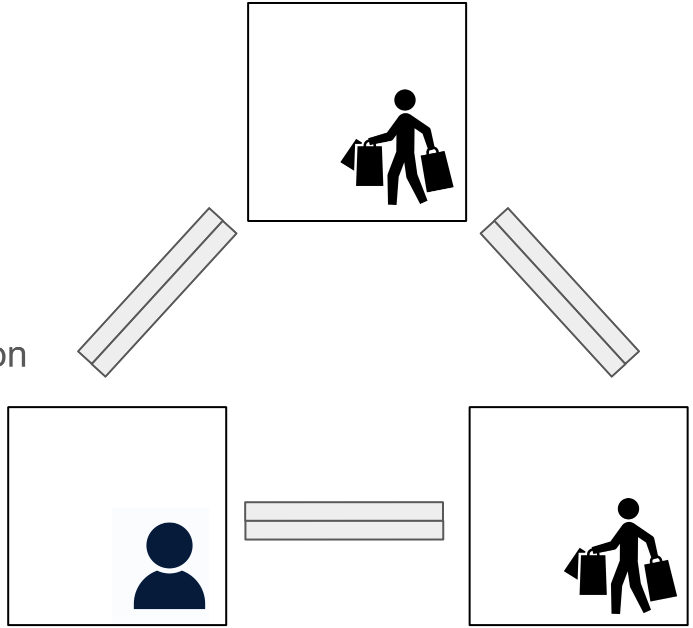
- High-level message sequence charts [Mauw and Reniers 97]
- Session types [Honda et al. 08]
- Choreographic programming [Carbone and Montesi 13]

Motivation

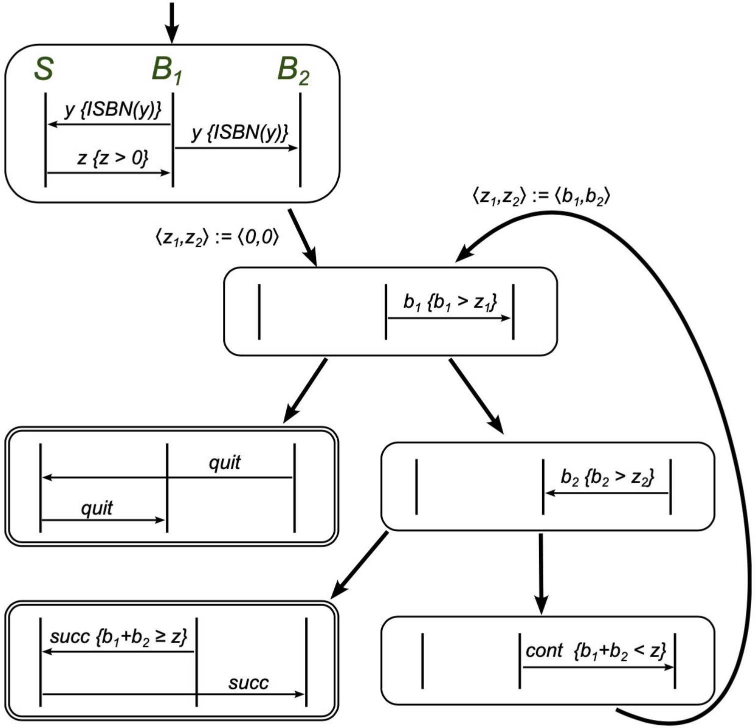




correct-by-construction
synthesis

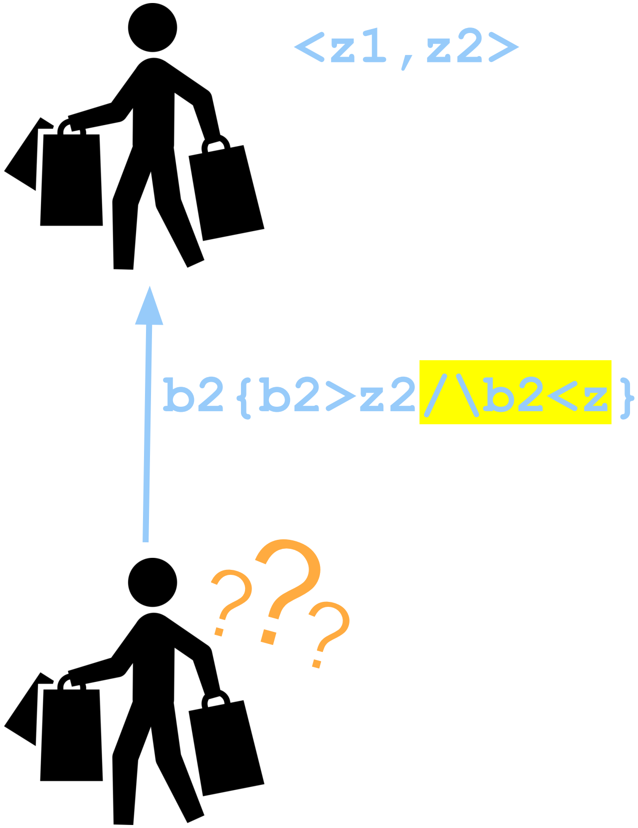


Motivation


S
B1
B2
y {ISBN(y)}
y {ISBN(y)}
z {z > 0}
⟨z1,z2⟩ := ⟨0,0⟩
⟨z1,z2⟩ := ⟨b1,b2⟩
b1 {b1 > z1}
quit
quit
b2 {b2 > z2}
succ {b1+b2 ≥ z}
succ
cont {b1+b2 < z}


?
implementability

Non-implementable two-bidder protocol



Coherence Conditions [Li et al. OOPSLA'25]

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

In a nutshell:

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)

Coherence Conditions [Li et al. OOPSLA'25]

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

In a nutshell:

μCLP formulae
(fixpoint logic modulo first-order theories)

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)

sound and complete μCLP reduction for implementability of symbolic protocols

SPROUT

Input

```
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)
```

↓

Symbolic Coherence Conditions [Li et al. OOPSLA'25]

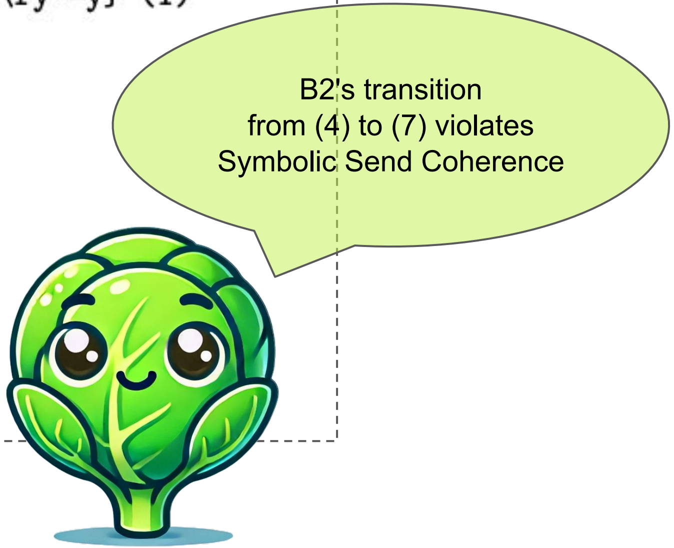
↓

MuVal [Unno et al. 23]

✓ ✗

Reporting implementability violations

```
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)
```



Expressivity

[Zhou et al. 20]

Example	$\lvert\mathcal{P}\rvert$	Impl.	SPROUT	Time	SESSION*	Time
Calculator	2	✓	✓	0.6s	N/A	2.0s
Fibonacci	2	✓	✓	0.5s	✓	1.8s
HigherLower	3	✓	✓	15.2s	✓	3.9s
HTTP	2	✓	✓	0.4s	✓	1.9s
Negotiation	2	✓	✓	1.0s	✓	1.9s
OnlineWallet	3	✓	✓	9.4s	✓	3.3s
SH	3	✓	✓	237.1s	✓	5.6s
Ticket	2	✓	✓	0.6s	✓	1.9s
TravelAgency	2	✓	✓	9.2s	✓	3.1s
TwoBuyer	3	✓	✓	3.8s	✓	2.8s
DoubleBuffering	3	✓	✓	1.5s	✓	2.3s
OAuth	3	✓	✓	6.2s	✓	2.3s
PlusMinus	3	✓	✓	5.2s	✗	2.1s
RingMax	7	✓	✓	3.7s	✓	4.7s
SimpleAuth	2	✓	✓	0.5s	✓	2.0s
TravelAgency2	2	✓	✓	1.7s	✓	1.8s

Expressivity

[Zhou et al. 20]

Example	$\|\mathcal{P}\|$	Impl.	SPROUT	Time	SESSION*	Time
Calculator	2	✓	✓	0.6s	N/A	2.0s
Fibonacci	2	✓	✓	0.5s	✓	1.8s
HigherLower	3	✓	✓	15.2s	✓	3.9s
HTTP	2	✓	✓	0.4s	✓	1.9s
Negotiation	2	✓	✓	1.0s	✓	1.9s
OnlineWallet	3	✓	✓	9.4s	✓	3.3s
SH	3	✓	✓	237.1s	✓	5.6s
Ticket	2	✓	✓	0.6s	✓	1.9s
TravelAgency	2	✓	✓	9.2s	✓	3.1s
TwoBuyer	3	✓	✓	3.8s	✓	2.8s
DoubleBuffering	3	✓	✓	1.5s	✓	2.3s
OAuth	3	✓	✓	6.2s	✓	2.3s
PlusMinus	3	✓	✓	5.2s	×	2.1s
RingMax	7	✓	✓	3.7s	✓	4.7s
SimpleAuth	2	✓	✓	0.5s	✓	2.0s
TravelAgency2	2	✓	✓	1.7s	✓	1.8s

Multiparty session types literature contains only positive examples!

Precision

[Zhou et al. 20]

Example	$\lvert\mathcal{P}\rvert$	Impl.	SPROUT	Time	SESSION*	Time
send-validity-yes	4	✓	✓	1.9s	✗	2.1s
send-validity-no	4	✗	✗	1.9s	✗	2.1s
receive-validity-yes	3	✓	✓	5.1s	✗	2.3s
receive-validity-no	3	✗	✗	3.6s	✗	2.0s
symbolic-two-bidder-yes	3	✓	✓	27.4s	✗	2.0s
symbolic-two-bidder-no1	3	✗	✗	30.0s	✗	2.1s
figure12-yes	3	✓	✓	2.0s	✓	2.0s
figure12-no	3	✗	✗	3.0s	✓	3.0s
symbolic-send-validity-yes	4	✓	✓	6.5s	✗	2.5s
symbolic-send-validity-no	4	✗	✗	5.3s	✗	2.6s
symbolic-receive-validity-yes	3	✓	✓	6.6s	✗	2.8s
symbolic-receive-validity-no	3	✗	✗	7.6s	✗	2.8s
fwd-auth-yes	3	✓	✓	10.3s	✗	2.3s
fwd-auth-no	3	✗	?	T/O	✗	2.2s
symbolic-two-bidder-no2	3	✗	✗	23.9s	✗	2.8s
higher-lower-ultimate	3	✓	✓	11.1s	✗	2.4s
higher-lower-winning	3	✓	?	T/O	✓	229.8s
higher-lower-no	3	✗	✗	7.3s	N/A	2.2s
higher-lower-encrypt-yes	4	✓	✓	9.3s	✗	2.3s
higher-lower-encrypt-no	4	✗	✗	177.3s	✗	2.4s
higher-lower-mixed	3	✗	✗	19.3s	✗	2.3s

Precision

[Zhou et al. 20]

Example	$\lvert\mathcal{P}\rvert$	Impl.	SPROUT	Time	SESSION*	Time
send-validity-yes	4	✓	✓	1.9s	×	2.1s
send-validity-no	4	×	×	1.9s	×	2.1s
receive-validity-yes	3	✓	✓	5.1s	×	2.3s
receive-validity-no	3	×	×	3.6s	×	2.0s
symbolic-two-bidder-yes	3	✓	✓	27.4s	×	2.0s
symbolic-two-bidder-no1	3	×	×	30.0s	×	2.1s
figure12-yes	3	✓	✓	2.0s	✓	2.0s
figure12-no	3	×	×	3.0s	✓	3.0s
symbolic-send-validity-yes	4	✓	✓	6.5s	×	2.5s
symbolic-send-validity-no	4	×	×	5.3s	×	2.6s
symbolic-receive-validity-yes	3	✓	✓	6.6s	×	2.8s
symbolic-receive-validity-no	3	×	×	7.6s	×	2.8s
fwd-auth-yes	3	✓	✓	10.3s	×	2.3s
fwd-auth-no	3	×	?	T/O	×	2.2s
symbolic-two-bidder-no2	3	×	×	23.9s	×	2.8s
higher-lower-ultimate	3	✓	✓	11.1s	×	2.4s
higher-lower-winning	3	✓	?	T/O	✓	229.8s
higher-lower-no	3	×	×	7.3s	N/A	2.2s
higher-lower-encrypt-yes	4	✓	✓	9.3s	×	2.3s
higher-lower-encrypt-no	4	×	×	177.3s	×	2.4s
higher-lower-mixed	3	×	×	19.3s	×	2.3s

Efficiency

[Zhou et al. 20]

Example	$\lvert\mathcal{P}\rvert$	Impl.	SPROUT	Time	SESSION*	Time
send-validity-yes	4	✓	✓	1.9s	✗	2.1s
send-validity-no	4	✗	✗	1.9s	✗	2.1s
receive-validity-yes	3	✓	✓	5.1s	✗	2.3s
receive-validity-no	3	✗	✗	3.6s	✗	2.0s
symbolic-two-bidder-yes	3	✓	✓	27.4s	✗	2.0s
symbolic-two-bidder-no1	3	✗	✗	30.0s	✗	2.1s
figure12-yes	3	✓	✓	2.0s	✓	2.0s
figure12-no	3	✗	✗	3.0s	✓	3.0s
symbolic-send-validity-yes	4	✓	✓	6.5s	✗	2.5s
symbolic-send-validity-no	4	✗	✗	5.3s	✗	2.6s
symbolic-receive-validity-yes	3	✓	✓	6.6s	✗	2.8s
symbolic-receive-validity-no	3	✗	✗	7.6s	✗	2.8s
fwd-auth-yes	3	✓	✓	10.3s	✗	2.3s
fwd-auth-no	3	✗	?	T/O	✗	2.2s
symbolic-two-bidder-no2	3	✗	✗	23.9s	✗	2.8s
higher-lower-ultimate	3	✓	✓	11.1s	✗	2.4s
higher-lower-winning	3	✓	?	T/O	✓	229.8s
higher-lower-no	3	✗	✗	7.3s	N/A	2.2s
higher-lower-encrypt-yes	4	✓	✓	9.3s	✗	2.3s
higher-lower-encrypt-no	4	✗	✗	177.3s	✗	2.4s
higher-lower-mixed	3	✗	✗	19.3s	✗	2.3s

Conclusion

- Integration
- Metatheory mechanized in Rocq [Li and Wies ITP'25]
- Extensions:
  - Property checking
  - Other asynchronous network architectures
  - Performance improvement via parallelization
- Future work: synthesis of symbolic implementations
- Optimization and MuVal bugs see CAV'25, incompleteness analysis see OOPSLA'25

https://github.com/nyu-acsys/sprout/

https://efl9013.github.io/

Conclusion

CAV Artifact Evaluation ★ Available

CAV Artifact Evaluation ★★★ Reusable

- Integration
- Metatheory mechanized in Rocq [Li and Wies ITP'25]
- Extensions:
  - Property checking
  - Other asynchronous network architectures
  - Performance improvement via parallelization
- Future work: synthesis of symbolic implementations
- Optimization and MuVal bugs see CAV'25, incompleteness analysis see OOPSLA'25

https://github.com/nyu-acsys/sprout/

https://efl9013.github.io/

Thank you!