TOWARDS A COMPOSITIONAL SESSION LOGIC FOR COMMUNICATION PROTOCOLS

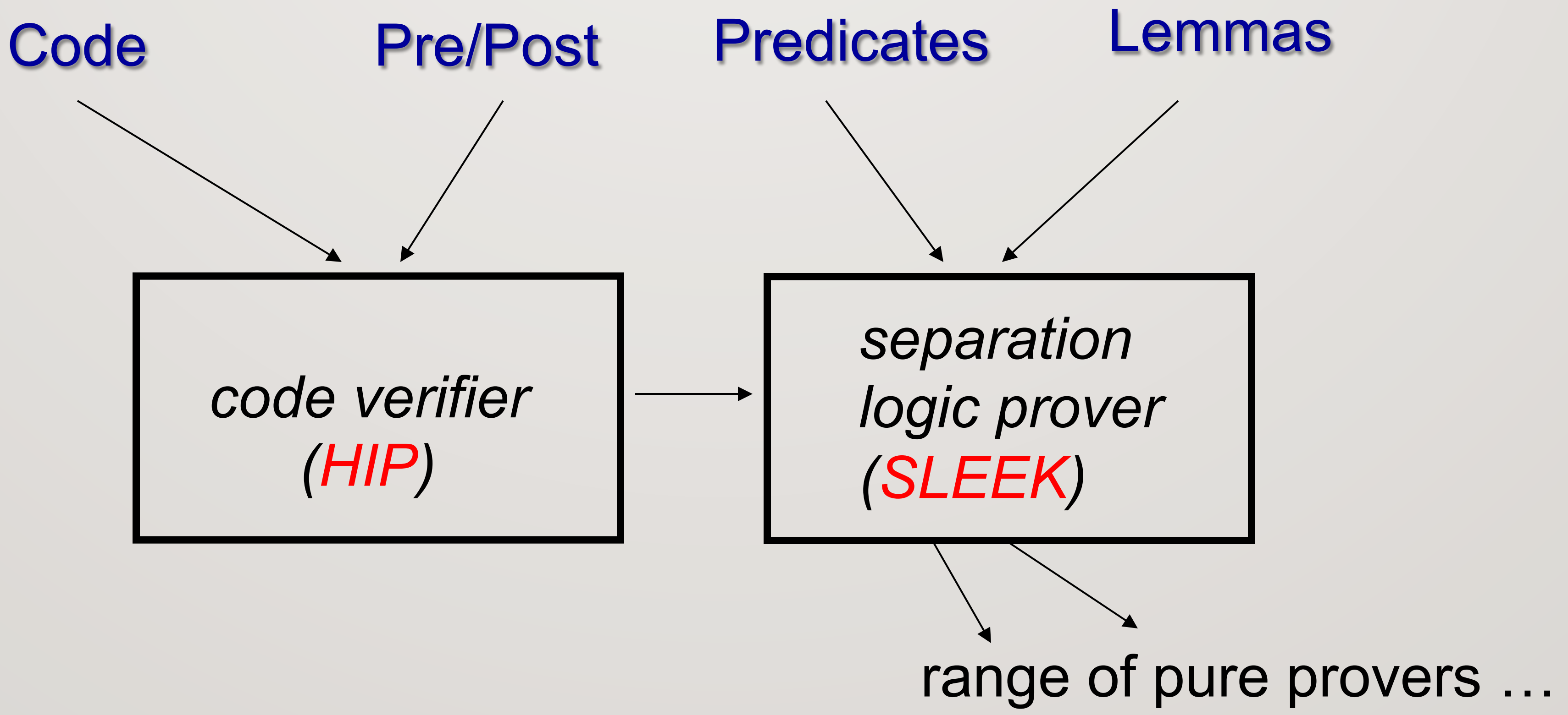
@Northern Concurrency 2017

Andreea Costea, Florin Craciun, Shengchao Qin, Wei-Ngan Chin

School Of Computing, National University Of Singapore

PROJECT

Under development since 2006 (200K lines of OCaml)
Currently : 3 current PhD students; 7 graduated PhD



Omega, MONA, Isabelle, Coq, SMT, Redlog, MiniSAT, Mathematica

FEATURES OF HIP/SLEEK

Can specify *complex* data structures to support symbolic verification.

(i) *expressive* (shapes+size, term, //ism)

(ii) *automation* (with lemma, inference)

(iii) *modular* (per method)

(iv) more *scalable* (proof slicing)

http://loris-5.d2.comp.nus.edu.sg/Tut1/

TOWARDS A COMPOSITIONAL SESSION LOGIC FOR COMMUNICATION PROTOCOLS

COMMUNICATION PROTOCOLS

Are everywhere: OS, parallel computation, transportation system, etc.

But,

writing such software is error-prone!

Therefore,

need support to formally specify and verify their correctness.

COMMUNICATION PROTOCOLS

Session type theory[1,2,3]

- intensively studied: maturity (multiparty, delegation, hybrid solutions)
- correct message type sequencing
- deadlock freedom, progress and liveness

But, it should also be important to:

- go beyond types and reason about the content of the message (value, resource)
- support specialized protocols in a general context
- add fewer constraints over the input language

1. K. Honda, N. Yoshida, M. Carbone. Multiparty Asynchronous Session Types. In POPL 2008
2. D. Orchard No. Yoshida. Effects as Sessions, Sessions as Effects. In POPL 2016
3. M. Carbone, S. Lindley, F. Montesi, C. Shürmann, and P. Wadler. Coherence generalises duality: a logical explanation of multiparty session types. In CONCUR 2016.

TOWARDS A SESSION LOGIC FOR COMMUNICATION PROTOCOLS

KEY ISSUES

Type safety → Logic which treats communication as resource

Variance or Flow-Awareness

Higher-Order Predicates

Multiparty & Synchronization

COMMUNICATION ASSUMPTIONS

Bidirectional channels / Mailbox channel per party

Asynchronous send and blocking receive

Explicit Synchronization

SESSION LOGIC – BINARY PROTOCOLS – Operators (extension of Separation Logic)

P =

$?r \cdot \phi$: *input*

$!r \cdot \phi$: *output*

$P_1 ; P_2$: *sequence*

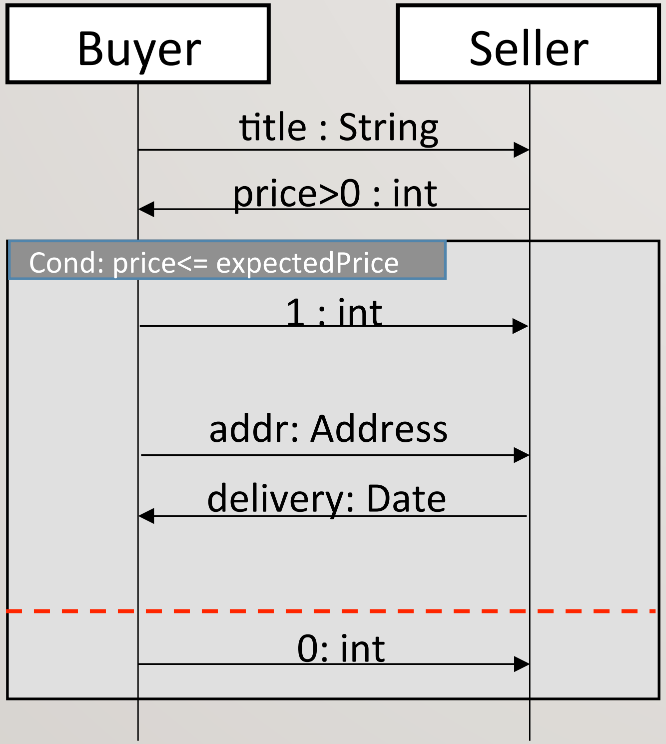
$P_1 \vee P_2$: *choice*

$P_1 * P_2$: *non-determistic*

Shape pred.	$spred$	$:= c\langle \mathtt{v}^* \rangle \equiv \phi \ \mathtt{inv} \ \pi$
Formula	ϕ	$:= \bigvee (\exists \mathtt{v}^* \cdot \kappa \wedge \pi)^*$
Pure formula	π	$:= \gamma \wedge \varphi$
Ptr. eq./diseq.	γ	$:= \mathtt{v}{=}\mathtt{v} \mid \mathtt{v}{=}\mathtt{null} \mid \mathtt{v}{\neq}\mathtt{v} \mid \mathtt{v}{\neq}\mathtt{null} \mid \gamma_1 \wedge \gamma_2$
Heap formula	κ	$:= \mathtt{emp} \mid \mathtt{v} {\mapsto} \mathtt{c}(\mathtt{v}^*) \mid \mathtt{p}\langle \mathtt{v}^* \rangle \mid \mathcal{C}\langle \mathtt{v}, \mathtt{P} \rangle \mid \kappa * \kappa$
	Δ	$::= \Phi \mid \Delta \vee \Delta \mid \Delta \wedge \pi \mid \Delta * \Delta \mid \exists \mathtt{v} \cdot \Delta$
	φ	$::= \varphi \mid b \mid c \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg \varphi \mid \exists \mathtt{v} \cdot \varphi \mid \forall \mathtt{v} \cdot \varphi$
	b	$:= \mathtt{true} \mid \mathtt{false} \mid \mathtt{v} \mid b{=}b$
	c	$:= \mathtt{a}{=}\mathtt{a} \mid \mathtt{a} {\leq} \mathtt{a}$
Presburger arith.	a	$:= \mathtt{k}^{\mathtt{int}} \mid \mathtt{v} \mid \mathtt{k}^{\mathtt{int}} \times \mathtt{a} \mid \mathtt{a}{+}\mathtt{a} \mid -\mathtt{a}$

$\mathcal{C}$(v, P) – associates spec P to channel v

SESSION LOGIC – BINARY PROTOCOLS – Example



a

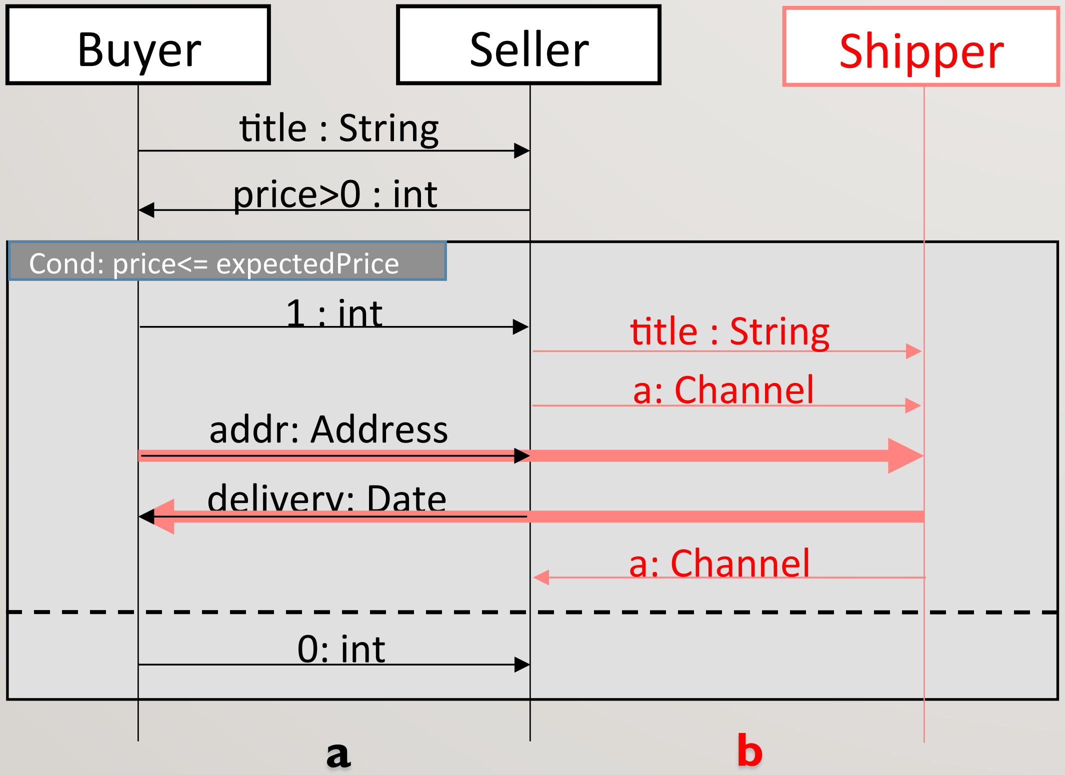
$!r \cdot emp \wedge r{:}String \wedge true$

$?r \cdot emp \wedge r{:}int \wedge r>0$

$$\mathtt{buyer_a} \triangleq\ !\mathtt{String};\ ?\mathtt{v} : \mathtt{int} \cdot \mathtt{v} > 0;\ ((!1;\ !\mathtt{Address};\ ?\mathtt{Date}) \vee\ !0)$$
$$\mathtt{seller_a} \triangleq\ ?\mathtt{String};\ !\mathtt{v} : \mathtt{int} \cdot \mathtt{v} > 0;\ ((?1;\ ?\mathtt{Address};\ !\mathtt{Date}) \vee\ ?0)$$

$!r \cdot emp \wedge r{:}int \wedge r=1$

SESSION LOGIC – BINARY PROTOCOLS – delegation



$$\mathtt{buyer_a} \triangleq \mathtt{!String}; \mathtt{?v} : \mathtt{int} \cdot \mathtt{v} > 0; ((\mathtt{!1}; \mathtt{!Address}; \mathtt{?Date}) \vee \mathtt{!0})$$

$$\mathtt{seller_a} \triangleq \mathtt{?String}; \mathtt{!v} : \mathtt{int} \cdot \mathtt{v} > 0; ((\mathtt{?1}; \mathtt{?Address}; \mathtt{!Date}) \vee \mathtt{?0})$$

$$\mathtt{seller_b} \triangleq \mathtt{!String}; \mathtt{!v} \cdot \mathcal{C}(\mathtt{v}, \mathtt{?Address}; \mathtt{!Date}); \mathtt{?v} \cdot \mathcal{C}(\mathtt{v}, \mathtt{emp})$$

$$\mathtt{shipper_b} \triangleq \mathtt{?String}; \mathtt{?v} \cdot \mathcal{C}(\mathtt{v}, \mathtt{?Address}; \mathtt{!Date}); \mathtt{!v} \cdot \mathcal{C}(\mathtt{v}, \mathtt{emp})$$

PLUGGING THE LOGIC INTO A SOFTWARE VERIFIER[4]

4. W. N. Chin, C. David, H.H. Nguyen, S. Qin. Automated verification of shape, size and bag properties via user-defined predicates in separation logic. *Science of Computer Programming,* Volume 77, Number 9, August 2012

SESSION LOGIC – Selected Entailment Rules

$$\frac{P_a \vdash P_c \;\rightsquigarrow\; S' \qquad S{=}\{\pi_i^e | \pi_i^e{\in}S'\}}{\mathcal{C}(v, P_a) \vdash \mathcal{C}(v, P_c) \;\rightsquigarrow\; S}\;[\mathbf{ENT{-}CHAN}]$$

$$\frac{\Box_a \vdash \Box_c \;\rightsquigarrow\; S_1 \qquad P_a \vdash P_c \;\rightsquigarrow\; S_2 \qquad \text{where } \Box := ?v \cdot \Delta \mid !v \cdot \Delta \mid \zeta}{\Box_a ; P_a \vdash \Box_c ; P_c \;\rightsquigarrow\; \{\mathtt{emp}{\wedge}\pi_1{\wedge}\pi_2 \mid \pi_1{\in}S_1{\wedge}\pi_2{\in}S_2\}}\;[\mathbf{ENT{-}SEQ}]$$

$$\frac{P_i ; P_a \vdash P_c \;\rightsquigarrow\; S_i \qquad S = \{\bigvee_i \Delta_i \mid \Delta_i{\in}S_i\}}{(\bigvee_i P_i) ; P_a \vdash P_c \;\rightsquigarrow\; S}\;[\mathbf{ENT{-}LHS{-}OR}]$$

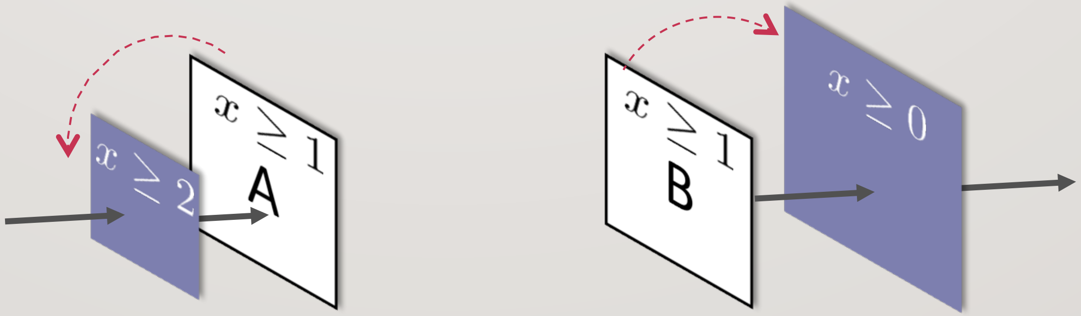
$$\frac{\Delta_a \vdash \Delta_c \;\rightsquigarrow\; S' \qquad S{=}\{\pi_i^e | \pi_i^e{\in}S'\}}{?v \cdot \Delta_a \vdash ?v \cdot \Delta_c \;\rightsquigarrow\; S}\;[\mathbf{ENT{-}RECV}]$$

$$\frac{\Delta_c \vdash \Delta_a \;\rightsquigarrow\; S' \qquad S{=}\{\pi_i^e | \pi_i^e{\in}S'\}}{!v \cdot \Delta_a \vdash !v \cdot \Delta_c \;\rightsquigarrow\; S}\;[\mathbf{ENT{-}SEND}]$$

SESSION LOGIC – BINARY PROTOCOLS – Variance

Example:

A: $\mathcal{C}(v, !x \cdot x \geq 1)$ || B: $\mathcal{C}(v, ?x \cdot x \geq 1)$



$!x \cdot x \geq 1 \vdash !x \cdot x \geq 2$

contravariant

$?x \cdot x \geq 1 \vdash ?x \cdot x \geq 0$

covariant

COMMUNICATION PRIMITIVES

$$\boxed{\textbf{FV}-\textbf{OPEN}}$$

$$\vdash \{\mathtt{emp}\}\ \mathtt{open}(\mathtt{c})\ \mathtt{with}\ \mathtt{P}\ \{\mathcal{C}(\mathtt{c}, \mathtt{S}_1) * \mathcal{C}(\mathtt{c}, \mathtt{S}_2)\}$$

$$\boxed{\textbf{FV}-\textbf{CLOSE}}$$

$$\vdash\ \{\mathcal{C}(\mathtt{c}, \mathtt{emp}) * \mathcal{C}(\mathtt{c}, \mathtt{emp})\}\ \mathtt{close}(\mathtt{c})\ \{\mathtt{emp}\}$$

$$\boxed{\textbf{FV}-\textbf{SEND}}$$

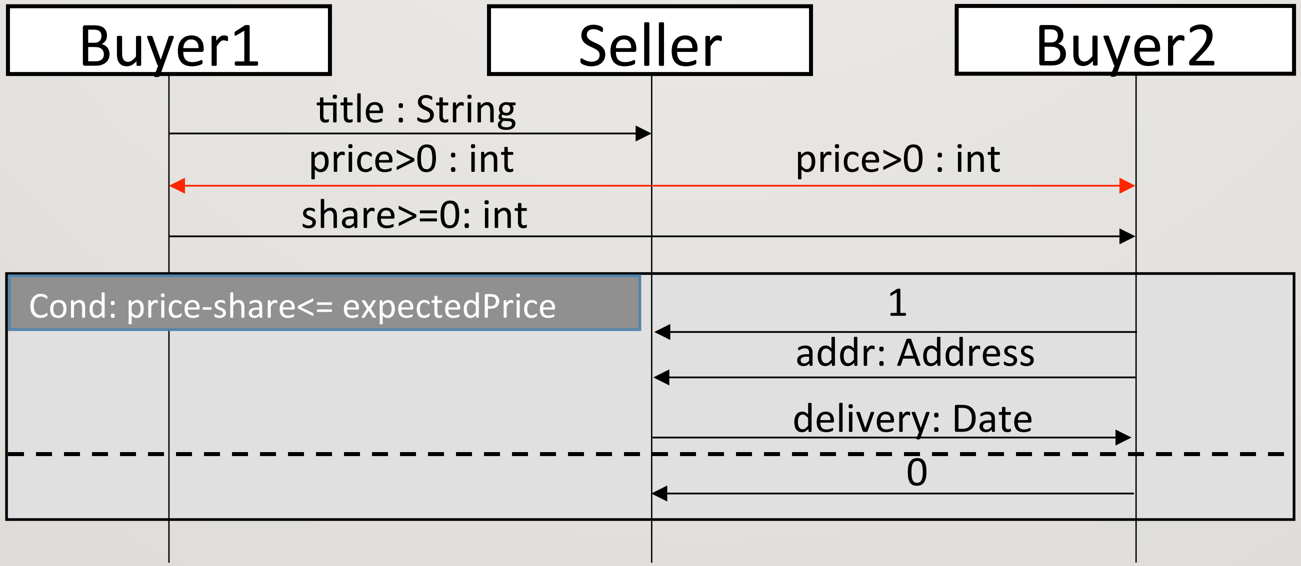
$$\vdash\ \{\mathcal{C}(\mathtt{c}, !\mathtt{v} \cdot \mathtt{L}(\mathtt{v}); \mathtt{P}) * \mathtt{L}(\mathtt{x})\}\ \mathtt{send}(\mathtt{c}, \mathtt{x})\ \{\mathcal{C}(\mathtt{c}, \mathtt{P})\}$$

$$\boxed{\textbf{FV}-\textbf{RECV}}$$

$$\vdash\ \{\mathcal{C}(\mathtt{c}, ?\mathtt{v} \cdot \mathtt{L}(\mathtt{v}); \mathtt{P})\}\ \mathtt{x} = \mathtt{receive}(\mathtt{c})\ \{\mathtt{L}(\mathtt{x}) * \mathcal{C}(\mathtt{c}, \mathtt{P})\}$$

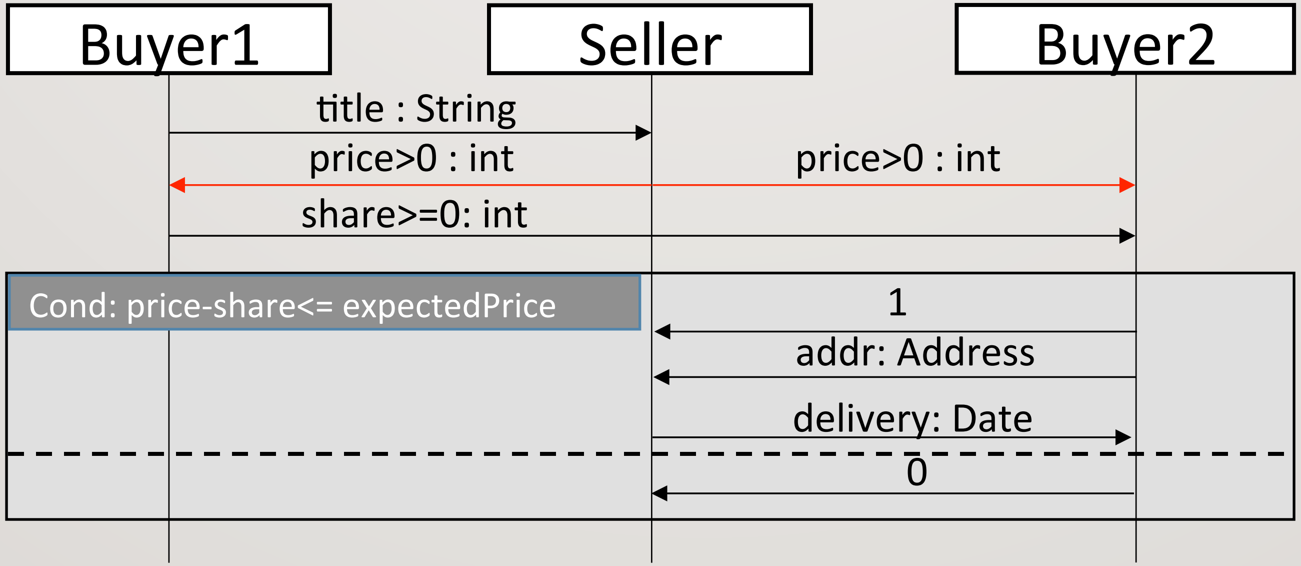
MULTI-PARTY SPECIFICATION

SESSION LOGIC – MULTIPARTY PROTOCOLS



$$G ::= A \to B : \Phi \mid A \xrightarrow{d} B : G \mid p(v^*) \mid \zeta_{id} \mid G \circledast G \mid G \vee G \mid G ; G$$

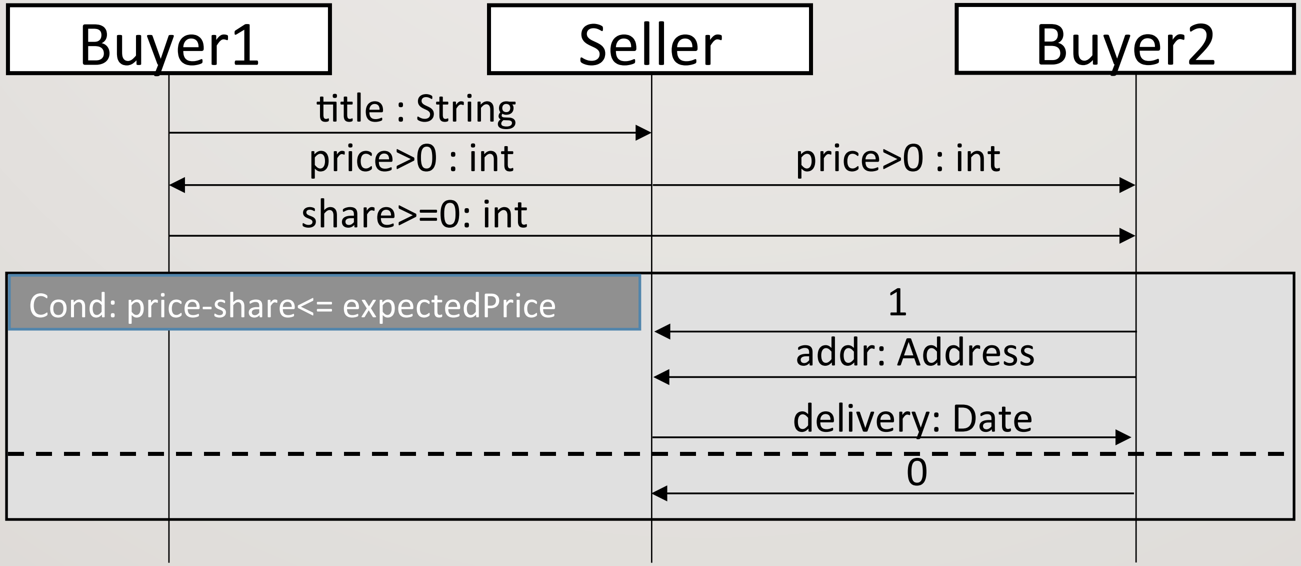
SESSION LOGIC – MULTIPARTY PROTOCOLS – Specs



arbitrary order

$$
\begin{aligned}
G_{BBS}(B_1, B_2, S) \triangleq\ & B_1 \to S : \mathtt{String}; \\
& ((S \to B_1 : v \cdot v > 0) \circledast (S \to B_2 : v \cdot v > 0)); \\
& B_2 \to B_1 : v \cdot v \geq 0; \\
& ((B_2 \to S : 1; B_2 \to S : \mathtt{Addr}; S \to B_2 : \mathtt{Date}) \vee (B_2 \to S : 0))
\end{aligned}
$$

SESSION LOGIC – MULTIPARTY PROTOCOLS – Specs (enforcing sequentiality between different channels)



$$
\begin{aligned}
G_{BBS}(B_1, B_2, S) \triangleq\ & \boxed{B_1 \to S} : \mathtt{String}; \zeta_1; \\
& ((\boxed{S \to B_1} : v \cdot v > 0) \circledast (\boxed{S \to B_2} : v \cdot v > 0)); \zeta_2; \\
& \boxed{B_2 \to B_1} : v \cdot v \geq 0; \zeta_3; \\
& ((\boxed{B_2 \to S} : 1; B_2 \to S : \mathtt{Addr}; S \to B_2 : \mathtt{Date}) \vee (\boxed{B_2 \to S} : 0))
\end{aligned}
$$

ζ_{id} - local or global sync instrument

SYNCHRONIZATION

IMPLICIT (NO) SYNCHRONIZATION

Consider: $A \rightarrow B : t_1$; $C \rightarrow D : t_2$

What are expected ordering?

$A\rightarrow$ before $\rightarrow B$

$C\rightarrow$ before $\rightarrow D$

EXPLICIT SYNCHRONIZATION

Consider: $A \rightarrow B : t_1 \quad ; \quad C \rightarrow D : t_2$

How about the other sequencing?

SS: $A\rightarrow$ before $C\rightarrow$ SR: $A\rightarrow$ before $\rightarrow D$

RS: $\rightarrow B$ before $C\rightarrow$ RR: $\rightarrow B$ before $\rightarrow D$

Which is the strongest? cf sequential consistency..

EXPLICIT SYNCHRONIZATION

Consider: $A \rightarrow B : t_1 \;;_{RS}\; C \rightarrow D : t_2$

Strict Sequencing RS: $\rightarrow B$ before $C \rightarrow$

Explicit enforcement:

		AWAIT(c)	
$A \rightarrow\ : t_1$	$\rightarrow B : t_1$	$C \rightarrow\ : t_2$	$\rightarrow D : t_2$
	DEC(c)		

STRICT SEQUENCING

Advantage: simpler to understand

Disadvantage: less concurrency;
need for explicit synchronization

MAILBOX COMMUNICATION

Consider: $A \rightarrow B : t_1 \;\; ; \;\; C \rightarrow B : t_2$

Assume mailbox communication + no synchronization

$A \rightarrow \, : t_1$ $\qquad$ $C \rightarrow \, : t_2$ $\qquad$ $\rightarrow B : t_1$

$\rightarrow B : t_2$

What if C sends before A to mailbox of B?
Type error in communication!

SEND-SEND SYNCHRONIZATION

Consider: $A \rightarrow B : t_1 \;\; ;_{SS} \;\; C \rightarrow B : t_2$

Enforce SS synchronization for mailbox communication

SS: $A \rightarrow$ before $C \rightarrow$

Explicit synchronization:

$A \rightarrow \; : t_1$	AWAIT(c)	$\rightarrow B : t_1$
DEC(c)	$C \rightarrow \; : t_2$	$\rightarrow B : t_2$

WITHOUT SYNCHRONIZATION

Consider: $A \rightarrow B : t_1 \;\; ; \;\; A \rightarrow C : t_2$

Decomposition without synchronization

$A \rightarrow\, : t_1$ $\quad\quad \rightarrow B : t_1 \quad\quad \rightarrow C : t_2$

$A \rightarrow\, : t_2$

What if B is our privileged customer whom we wish to notify ahead of our regular customer C?

RECEIVE-RECEIVE SYNCHRONIZATION

Consider: $A \rightarrow B : t_1 \;\; ;_{RR} \;\; A \rightarrow C : t_2$

Enforce RR synchronization for priority receive.

RR: $\rightarrow B$ before $C \rightarrow$

Explicit synchronization:

		AWAIT(c)
$A \rightarrow : t_1$	$\rightarrow B : t_1$	$\rightarrow C : t_2$
$A \rightarrow : t_2$	DEC(c)	

SYNCHRONIZATION

Both explicit and implicit synchronization can be used.

Priority (i) safety (ii) correctness (iii) user requirement (iv) performance.

Expressive logic + spec synthesis provide flexibility

TAKE AWAY

Type safety → Logic which treats communication as resource: concurrency, sequence, nondeterminism

Variance or Flow-Awareness: support for precise specification and verification

Higher-Order Predicates: expressive protocols

Multiparty → Locality: synchronization

Work in Progress

Synthesize local projections → Multiparty view

Explore more abstractions (e.g. specify security protocols)