Behavioral Types and Logical Frameworks
An Introduction

Carsten Schürmann
IT University of Copenhagen

carsten@demtech.dk

October 7, 2016

Motivation

Buzzwords

- Concurrency
- Linear Logic
- Delegation
- Services
- Security

Motivating Example

On a laptop not far from here ...

You want to buy a book form and online store, but only if the price is right.

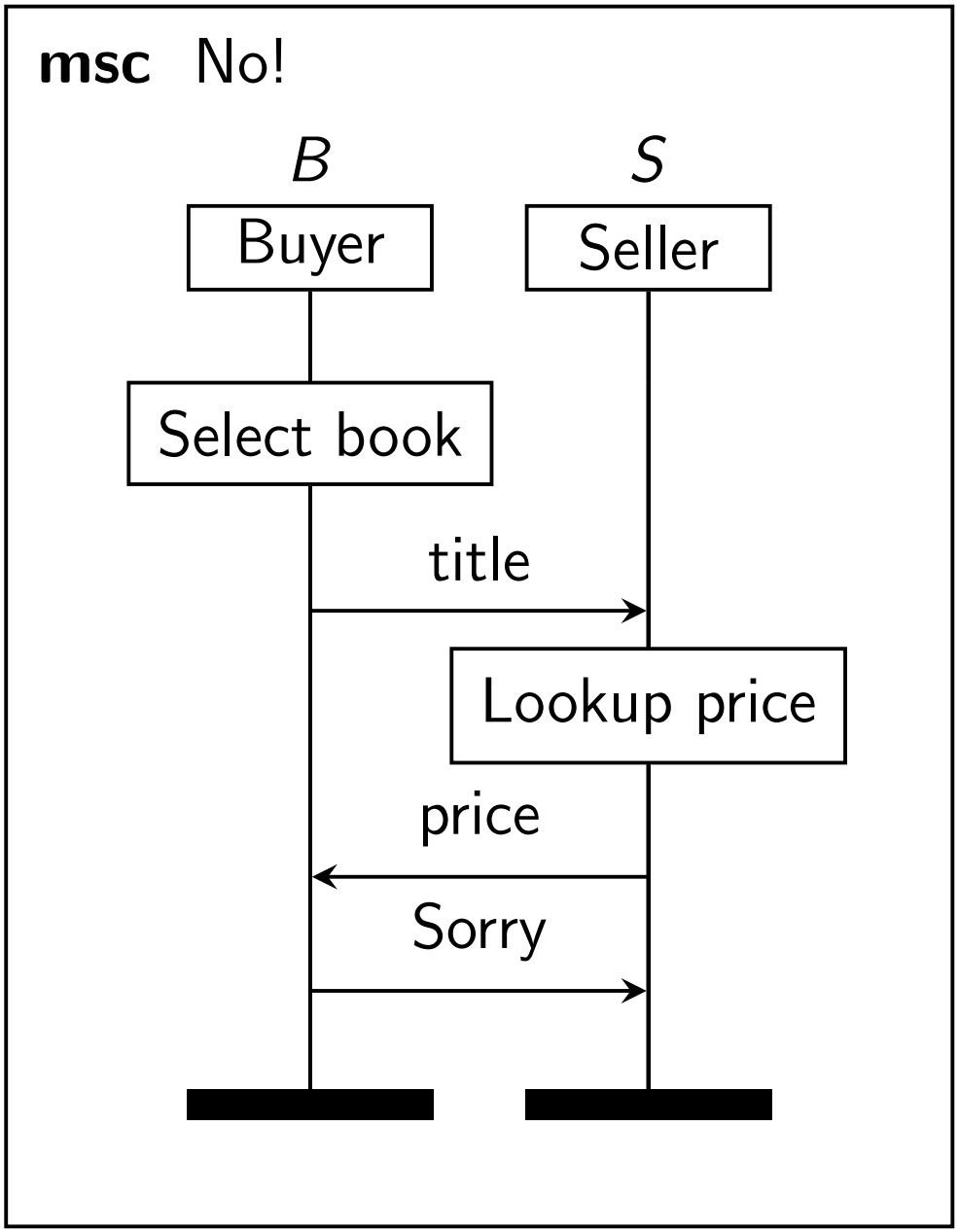
Motivating Example

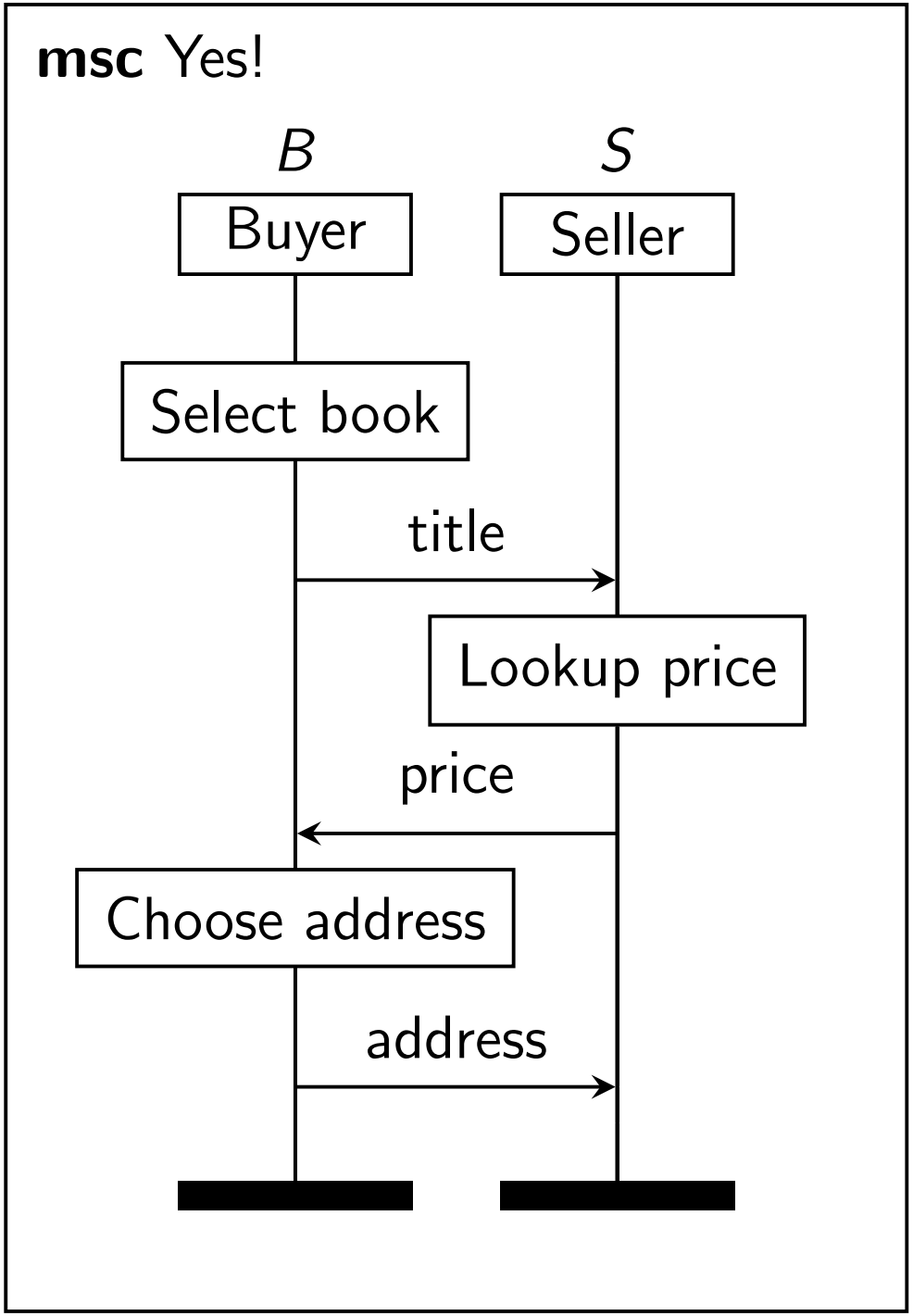
On a laptop not far from here ...

You want to buy a book form and online store, but only if the price is right.

Observations:

- More than one agent involved
- It is difficult to capture the invariant of such a system
- The "type" needs to capture the protocol of how messages are exchanged.


msc No!
B
S
Buyer
Seller
Select book
title
Lookup price
price
Sorry


msc Yes!
B
S
Buyer
Seller
Select book
title
Lookup price
price
Choose address
address

Activities

- BETTY Cost Action
- Session at POPL 2016
- Programming Language Design
  - SILL
  - Jolie

Ingredients

- Concurrency Theory [Caires, Carbone, Gay, Honda, Yoshida]
- Logic [Pfenning, CS, Toninho, Wadler]
- Programming Languages [Pfenning, Montesi, Toninho]

Functional Programming Semantics?

Observation 1

The protocol can be implemented in a functional language

```
fun lookup "Harry_Potter" = 45

fun buyerNo S =
      S (fn S' => fn price => S' (NONE))
        "Harry_Potter"
and buyerYes S =
      S (fn S' => fn price => S' (SOME "Berlin"))
        "Harry_Potter"
and seller B =
      fn title => B (fn NONE  => ()
          | SOME address => ()) (lookup title)
```

Logical Frameworks

The program satisfies the following types:

```
type B = ((('a option -> unit) -> int -> 'b)
          -> string -> 'd) -> 'd

type S = (('a option -> unit) -> int -> 'b)
         -> string -> 'b
```

Observation 2

- Judgments-as-types?
- Adequacy?
- Every message passing increases the order of the type
- Subtypes is duplicated multiple times
- Not even dependent types help, I suspect
- Conclusion: This is unwieldy

Alternative: Substructural Logical Frameworks

"$25 + 5 = 3 \times 10$"

Alternative: Substructural Logical Frameworks

"$25 + 5 = 3 \times 10$"

Alternative: Substructural Logical Frameworks

"$25 + 5 = 3 \times 10$"

Alternative: Substructural Logical Frameworks

"$25 + 5 = 3 \times 10$"

SSOS [Cervesato et al. '02]
SSOS [Pfenning, Simmons '13]
Celf [Schack-Nielsen, CS'11]

$$\texttt{R1} : \texttt{q} \multimap \{\texttt{d} \otimes \texttt{d} \otimes \texttt{n}\}.$$
$$\texttt{R2} : \texttt{n} \multimap \texttt{n} \multimap \{\texttt{d}\}.$$

Alternative: Substructural Logical Frameworks

"$25 + 5 = 3 \times 10$"

SSOS [Cervesato et al. '02]
SSOS [Pfenning, Simmons '13]
Celf [Schack-Nielsen, CS'11]

$$\texttt{R1} : \texttt{q} \multimap \{\texttt{d} \otimes \texttt{d} \otimes \texttt{n}\}.$$
$$\texttt{R2} : \texttt{n} \multimap \texttt{n} \multimap \{\texttt{d}\}.$$

Observation 3

- Multi-formula premisses
- Multi-formula conclusions
- Multi-set rewriting

The Concurrent World is Substructural

Substructural Logical Framework

Dependently typed language for multi-set rewriting rules

- $\otimes$ Connective to group facts
- $\multimap$ Connective to express rewrite rules
- $\exists$ To create new evidence of facts
- Π To quantify over evidence of facts

Substructural Operational Semantics

Multi-set rewriting semantics

- Forward-Chaining Search
- Runs until quiescence
- All truth is ephemeral

Motivation

The Central Questions of this Talk

What happens if we shift from a process algebra view of concurrency/session types to a purely logical view?
And how to do this shift?

1. Linear Logic
2. Session Types -as- Judgments
3. Adding Choice
4. Substructual Logical Framework CLF
5. Programming with Session Types
6. Demo
7. Conclusion and Future Work

Linear Logic

Judgmental reconstruction

Intuitionistic Logic

Logic of truth. .

Linear Logic

Logic of ephemeral resources. .

$$\Delta \vdash A$$

Judgmental reconstruction

Intuitionistic Logic

Logic of truth. (Logic of facts).

Linear Logic

Logic of ephemeral resources. (Logic of food).

$$\Delta \vdash A$$

Judgmental reconstruction

Intuitionistic Logic

Logic of truth. (Logic of facts).

Linear Logic

Logic of ephemeral resources. (Logic of food).

$$\Delta \vdash A$$

Important Properties:

- Cut-Elimination guarantees proof normalization
- Focusing, limits proofs but not provability
- Normal forms exist

Linear Logic – The Rules

$$\frac{}{\cdot \vdash 1}\ 1R \qquad \frac{\Delta \vdash C}{\Delta, 1 \vdash C}\ 1L$$

$$\frac{\Delta_1 \vdash A \quad \Delta_2 \vdash B}{\Delta_1, \Delta_2 \vdash A \otimes B}\ \otimes R \qquad \frac{\Delta, A, B \vdash C}{\Delta, A \otimes B \vdash C}\ \otimes L$$

$$\frac{\Delta, A \vdash B}{\Delta \vdash A \multimap B}\ \multimap R \qquad \frac{\Delta_1 \vdash A \quad \Delta_2, B \vdash C}{\Delta_1, \Delta_2, A \multimap B \vdash C}\ \multimap L$$

$$\frac{}{A \vdash A}\ \mathit{init}$$

$$\frac{\Delta_1 \vdash A \quad \Delta_2, A \vdash C}{\Delta_1, \Delta_2 \vdash C}\ \mathit{cut}$$

Linear Logic — Theorems

Goes back to ... [Girard '89]

Theorem (Admissibility of init)

For any formula A: $A \vdash A$.

Theorem (Admissibility of cut)

If $\Delta_1 \vdash A$ and $\Delta_2, A \vdash C$ then $\Delta_1, \Delta_2 \vdash C$.

Session Types -as- Judgments

Linear Logic - Primitive Types

- Making Linear Logic practical
- τ ranges over strings, integers, ...

$$\frac{\Gamma \vdash \quad \tau \text{ inhabited} \qquad \Gamma; \Delta \vdash \quad B}{\Gamma; \Delta \vdash \qquad\qquad \tau \wedge B} \wedge R$$

$$\frac{\Gamma, \quad \tau; \Delta, \quad B \vdash \quad C}{\Gamma; \Delta, \quad \tau \wedge B \vdash \qquad\qquad C} \wedge L$$

$$\frac{\Gamma, \quad \tau; \Delta \vdash \quad B}{\Gamma; \Delta \vdash \qquad\qquad \tau \supset B} \supset R$$

$$\frac{\Gamma \vdash \quad \tau \text{ inhabited} \qquad \Gamma; \Delta, \quad B \vdash \quad C}{\Gamma; \Delta, \quad \tau \supset B \vdash \qquad\qquad C} \supset L$$

Linear Logic - Primitive Types

- Making Linear Logic practical
- τ ranges over strings, integers, ...

$$\frac{\Gamma \vdash M : \tau \text{ inhabited} \quad \Gamma; \Delta \vdash T : B}{\Gamma; \Delta \vdash \text{send } \langle \text{M} \rangle; T : \tau \wedge B} \wedge R$$

$$\frac{\Gamma, x : \tau; \Delta, u : B \vdash T : C}{\Gamma; \Delta, u : \tau \wedge B \vdash \text{receive } (x)\ [u]; T : C} \wedge L$$

$$\frac{\Gamma, x : \tau; \Delta \vdash T : B}{\Gamma; \Delta \vdash \text{receive } (x); T : \tau \supset B} \supset R$$

$$\frac{\Gamma \vdash M : \tau \text{ inhabited} \quad \Gamma; \Delta, u : B \vdash T : C}{\Gamma; \Delta, u : \tau \supset B \vdash \text{send } \langle \text{M} \rangle\ [u]; T : C} \supset L$$

Encoding in a Substructural Logical Framework

[Pfenning and Griffith '15]

Terms T

1R	end
1L	wait $[u]$; T
$\wedge$R	send $\langle \mathsf{M} \rangle$; T
$\wedge$L	receive (x) $[u]$; T
$\supset$R	receive (x); T
$\supset$L	send $\langle \mathsf{M} \rangle$ $[u]$; T
$\otimes$R, $\otimes$L	...
$\multimap$R, $\multimap$L	...

Alternative: Use π-calculus to describe these processes.
[Caires & Pfenning '10, Wadler '12]

Related: Classical version of linear logic [Wadler'12]

Session Typing our Buyer Seller Example

$$\underset{\vdash \mathit{string} \wedge (\mathit{nat} \supset 1)}{B} \qquad \underset{(\mathit{string} \wedge (\mathit{nat} \supset 1)) \multimap 1 \vdash 1}{S}$$

Comments

- B aka buyer
- S aka seller
- Denote the derivation of the judgment

Buyer and Seller Example

$$\underset{\vdash \mathit{string} \wedge (\mathit{nat} \supset 1)}{B} \qquad \underset{\mathrm{u}:(\mathit{string} \wedge (\mathit{nat} \supset 1)) \multimap 1 \vdash 1}{S}$$

$$\begin{array}{lllll}
\mathit{Buyer} & B & = & \mathrm{send}\ \langle \text{"Harry_potter"} \rangle; \\
 & & & \mathrm{receive}\ (\mathit{price}); \\
 & & & \mathrm{end} \\
\mathit{Seller} & S\ [u] & = & \mathrm{receive}\ (\mathit{title})\ [u]; \\
 & & & \mathrm{send}\ \langle \$45 \rangle\ [u]; \\
 & & & \mathrm{wait}\ [u]; \\
 & & & \mathrm{end} \\
\mathit{System} & C & = & \mathrm{cut}\ B\ (S\ [u]).
\end{array}$$

Adding Choice

Linear Logic – The Additives

$$\frac{\Delta \vdash A_1 \quad \Delta \vdash A_2}{\Delta \vdash A_1 \& A_2}\ \&R$$

$$\frac{\Delta, A_1 \vdash C}{\Delta, A_1 \& A_2 \vdash C}\ \&L_1$$

$$\frac{\Delta, A_2 \vdash C}{\Delta, A_1 \& A_2 \vdash C}\ \&L_2$$

$$\frac{\Delta \vdash A_1}{\Delta \vdash A_1 \oplus A_2}\ \oplus R_1 \qquad \frac{\Delta \vdash A_2}{\Delta \vdash A_1 \oplus A_2}\ \oplus R_2$$

$$\frac{\Delta, A_1 \vdash C \quad \Delta, A_2 \vdash C}{\Delta, A_1 \oplus A_2 \vdash C}\ \oplus L$$

Linear Logic – The Additives

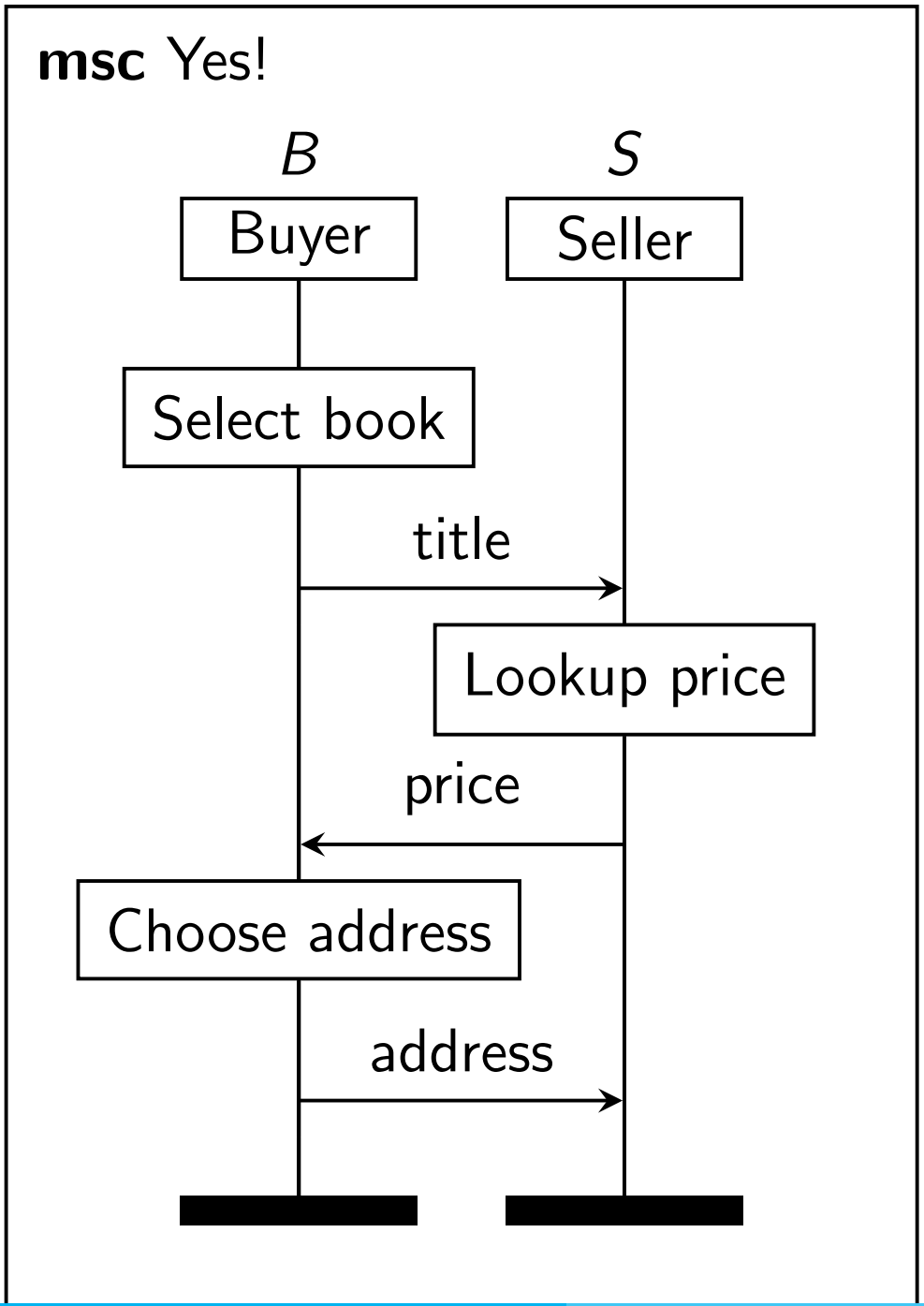
$$\frac{\Delta \vdash T_1 : A_1 \quad \Delta \vdash T_2 : A_2}{\Delta \vdash \mathsf{offer}\ (\mathsf{left} \Rightarrow T_1, \mathsf{right} \Rightarrow T_2) : A_1 \& A_2}\ \& R$$

$$\frac{\Delta, u : A_1 \vdash T : C}{\Delta, u : A_1 \& A_2 \vdash \mathsf{left}\ [u]; T : C}\ \& L_1$$

$$\frac{\Delta, u : A_2 \vdash T : C}{\Delta, u : A_1 \& A_2 \vdash \mathsf{right}\ [u]; T : C}\ \& L_2$$

$$\frac{\Delta \vdash T : A_1}{\Delta \vdash \mathsf{left}; T : A_1 \oplus A_2}\ \oplus R_1 \qquad \frac{\Delta \vdash T : A_2}{\Delta \vdash \mathsf{right}; T : A_1 \oplus A_2}\ \oplus R_2$$

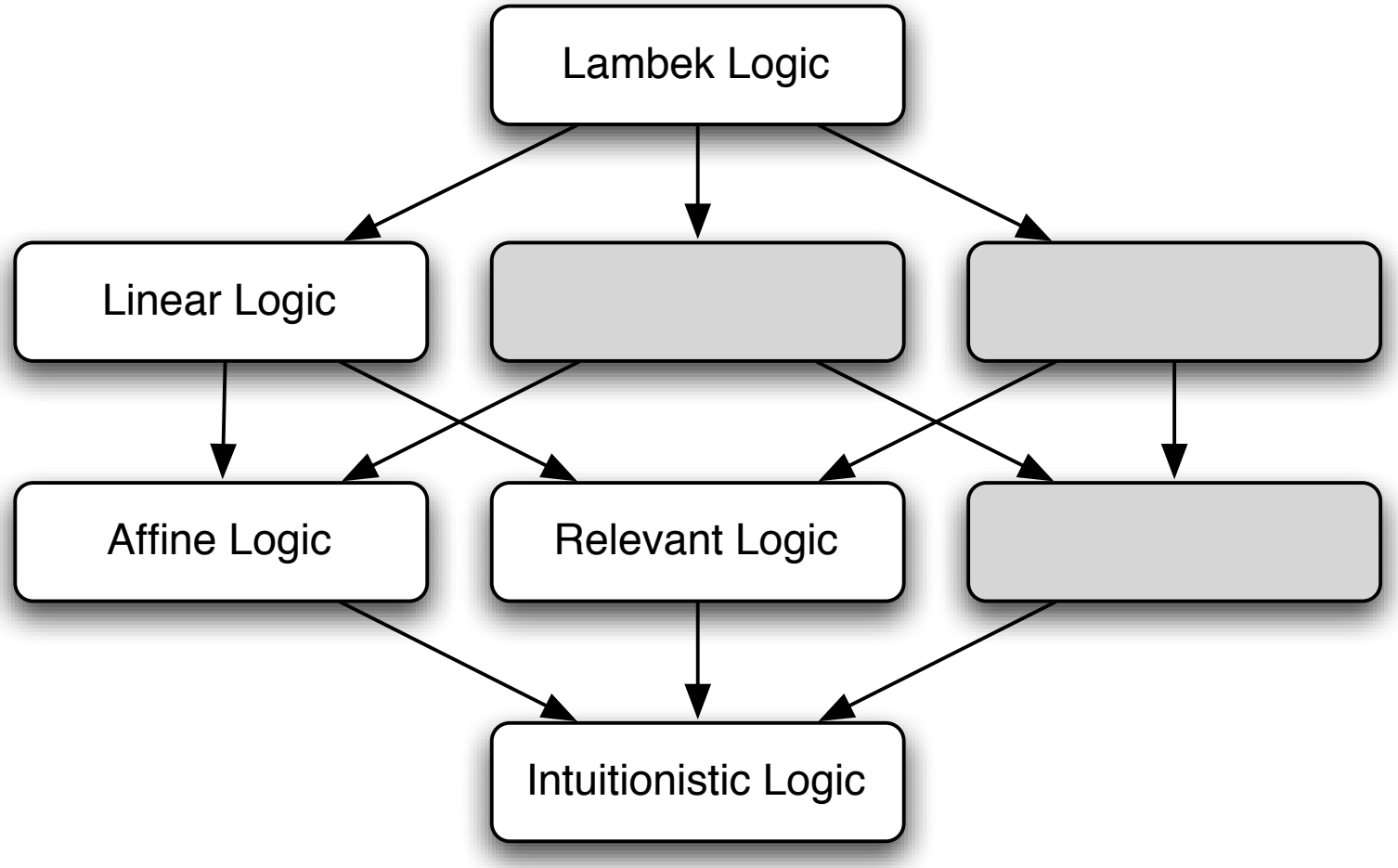
$$\frac{\Delta, v : A_1 \vdash T_1 : C \quad \Delta, w : A_2 \vdash T_2 : C}{\Delta, u : A_1 \oplus A_2 \vdash \mathsf{offer}\ [u]\ (\mathsf{left} \Rightarrow [v]; T_1, \mathsf{right} \Rightarrow [w]; T_2) : C}\ \oplus L$$



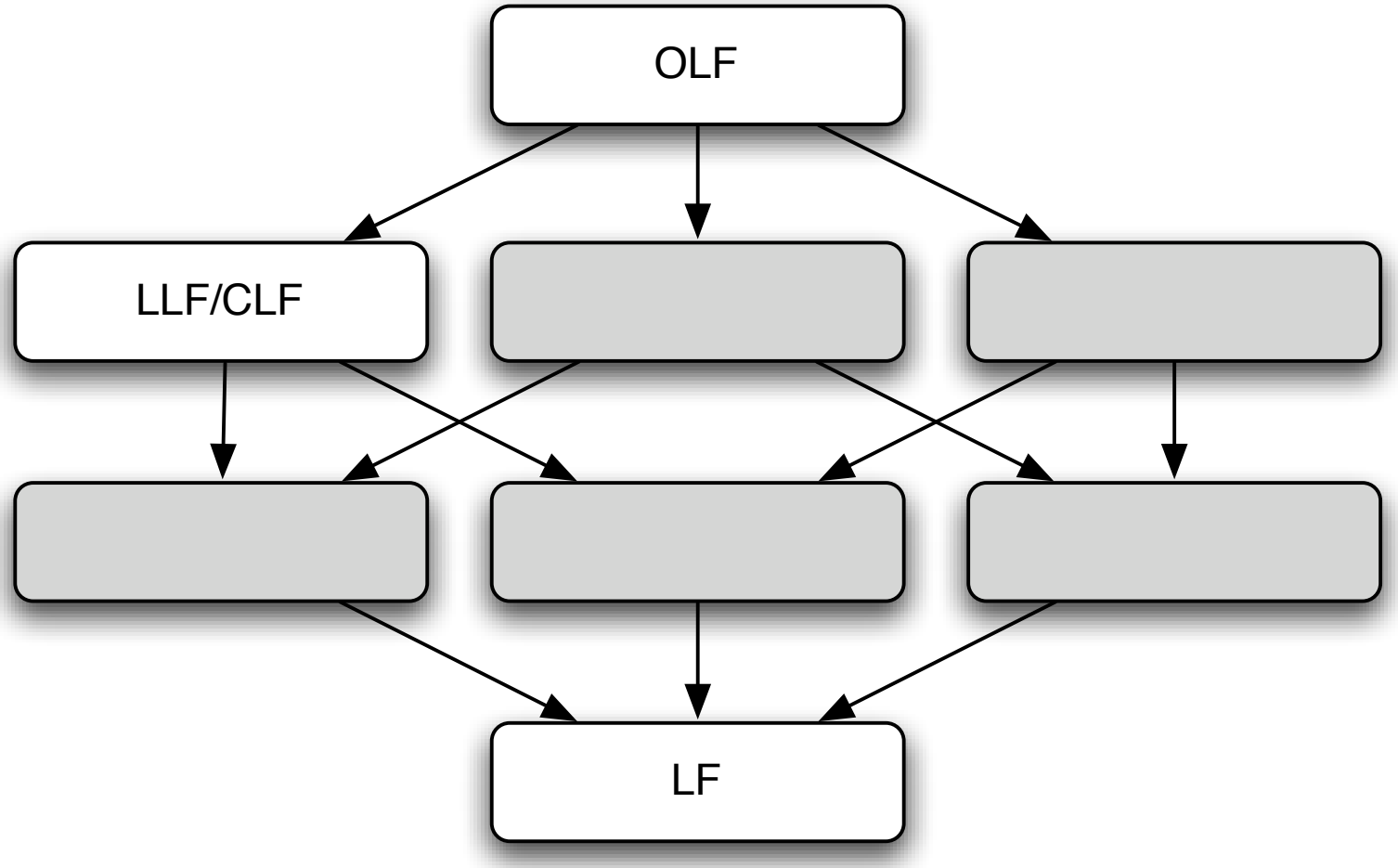
$$
\begin{aligned}
B = \quad & \mathsf{send}\ \langle \text{"Harry\_Potter"} \rangle; \\
& \mathsf{receive}\ (\mathit{price}); \\
& \mathsf{left}; \\
& \mathsf{send}\ (\text{"}\mathit{Berlin}\text{"}); \\
& \mathsf{end} \\
S = \quad & \mathsf{receive}\ (\mathit{title})\ [u]; \\
& \mathsf{send}\ \langle \$45 \rangle\ [u]; \\
& \mathsf{offer}[u](\\
& \mathsf{left} \Rightarrow \\
& \quad \mathsf{receive}\ (\mathit{address})\ [u]; \\
& \quad \mathsf{wait}\ [u]; \\
& \quad \mathsf{end} \\
& \mathsf{right} \Rightarrow \\
& \quad \mathsf{wait}\ [u]; \\
& \quad \mathsf{end})
\end{aligned}
$$

Substructual Logical Framework CLF

Substructural Logics



Substructural Logical Frameworks



Substructural Logics

$$\frac{A_1, \ldots, A_m}{B_1, \ldots, B_n}\ \mathit{name}$$

- In LLF order matters [Girard '89, Cervesato et al '96]

$$\mathit{name} : A_1 \otimes \cdots \otimes A_m \multimap B_1 \otimes \cdots \otimes B_n$$

- In CLF order does not matter [Cervesato et al '02]

$$\mathit{name} : A_1 \otimes \cdots \otimes A_m \multimap \{B_1 \otimes \cdots \otimes B_n\}$$

Execution as Proof Search

- Proof search

$$\begin{array}{c} A \\ \vdots \\ B \end{array}$$

  corresponds to inhabitation of types.

$$A \multimap \{B\}$$

- All terms are equal modulo interleavings
- No leftovers in the multi-set allowed
- Lollimon [Lopez et al. '05]
- Focusing [Andreoli '92, Chaudhuri '06, Miller '05]

Logical Framework CLF

- Focused version of Linear Logic [Andreoli '92]
- Conservative Extension of LF [Honsell, Harper, Plotkin '93]
- Types:

$$A ::= P \mid S \multimap A \mid \Pi x : S.\ A \mid A_1 \,\&\, A_2 \mid \{S\}$$
$$P ::= a \mid P\ N$$
$$S ::= 1 \mid S_1 \otimes S_2 \mid !A \mid @A \mid A \mid \exists x : S_1.\ S_2$$

- Kinds:

$$K ::= \mathsf{type} \mid \Pi x : A.\ K$$

We write $A \to B$ for $\Pi x : A.\ B$ if x does not occur in B.

CLF — Terms

Term syntax:

$$
\begin{array}{rll}
N ::= & \lambda p.\ N \mid \langle N_1, N_2 \rangle \mid \langle\rangle \mid \{E\} \mid & \\
& c \mid x \mid N_1\ N_2 \mid \pi_1\ N \mid \pi_2\ N & \textit{Objects} \\
E ::= & \text{let } \{p\} = N \text{ in } E \mid M & \textit{Expressions} \\
M ::= & M_1 \otimes M_2 \mid 1 \mid N \mid !N \mid @N \mid [N, M] & \textit{Monadic objects} \\
p ::= & p_1 \otimes p_2 \mid 1 \mid x \mid !x \mid @x \mid [x, p] & \textit{Patterns}
\end{array}
$$

Judgment

Let Γ unrestricted, Φ affine and Δ linear context.

$$\Gamma; \Phi; \Delta \vdash N : A$$

Equational Theory: α, β, η + let-floating

Judgments-as-types

$$\ulcorner \quad \begin{array}{c} P \\ A_1, \ldots, A_n \vdash C \end{array} \quad \urcorner$$

$$=$$

$$\cdot; u_1 : \mathsf{hyp}\ \ulcorner A_1 \urcorner, \ldots u_n : \mathsf{hyp}\ \ulcorner A_n \urcorner \vdash \ulcorner M \urcorner : \mathsf{conc}\ \ulcorner A \urcorner$$

Logical Framework Representation

```
o : type.                                  (Formulas)
conc : o -> type                          (Conclusions)
hyp : o -> type                           ( Hypotheses)
```

Example

```
buyer : conc (and string (imp nat one))
      = send "Harry_Potter" (
        receive  λ!price.
        end).

seller : hyp (and string (imp nat one)) -o conc one
       = λu. receive  (λ!title. λv.
             send $45 (λw.
             wait end w) v) u.
```

Programming with Session Types

Cut-Elimination

$$\dfrac{\dfrac{\dfrac{P}{\Delta, A \vdash B}}{\Delta \vdash A \multimap B}\multimap R \qquad \dfrac{\dfrac{Q_1}{\Delta_1 \vdash A} \qquad \dfrac{Q_2}{\Delta_2, B \vdash C}}{\Delta_1, \Delta_2, A \multimap B \vdash C}\multimap L}{\Delta, \Delta_1, \Delta_2 \vdash C}\,cut$$

Cut-Elimination

$$\dfrac{\dfrac{\dfrac{P}{\Delta, A \vdash B}}{\Delta \vdash A \multimap B}\multimap R \qquad \dfrac{Q_1}{\Delta_1 \vdash A} \qquad \dfrac{Q_2}{\Delta_2, B \vdash C}}{\Delta_1, \Delta_2, A \multimap B \vdash C}\multimap L}{\Delta, \Delta_1, \Delta_2 \vdash C}\,cut$$

reduces to

Cut-Elimination

$$\dfrac{\dfrac{\dfrac{P}{\Delta, A \vdash B}}{\Delta \vdash A \multimap B}\multimap R \qquad \dfrac{\dfrac{Q_1}{\Delta_1 \vdash A} \qquad \dfrac{Q_2}{\Delta_2, B \vdash C}}{\Delta_1, \Delta_2, A \multimap B \vdash C}\multimap L}{\Delta, \Delta_1, \Delta_2 \vdash C}\,cut$$

reduces to

$$\dfrac{\dfrac{\dfrac{Q_1}{\Delta_1 \vdash A} \qquad \dfrac{P}{\Delta, A \vdash B}}{\Delta, \Delta_1 \vdash B}\,cut \qquad \dfrac{Q_2}{\Delta_2, B \vdash C}}{\Delta, \Delta_1, \Delta_2 \vdash C}\,cut$$

Multi-Set Reduction

$$\left\{ \begin{array}{ccc} P & Q_1 & Q_2 \\ \Delta, A \vdash B & \Delta_1 \vdash A & \Delta_2, B \vdash C \end{array} \right\}$$

$$\Longrightarrow^{*}$$

$$\vdots$$

$$\Longrightarrow^{*}$$

$$\left\{ \begin{array}{c} Q \\ \cdot \vdash 1 \end{array} \right\}$$

Processes: always on, always connected!

$$\left\{ \begin{array}{cc} Q_1 & P \\ \Delta_1 \vdash A \ , & \Delta, u : A \vdash B \end{array} \right\} \Longrightarrow \left\{ \begin{array}{c} R \\ \Delta_1, \Delta \vdash B \end{array} \right\}$$

Representation in CLF

```
proc : conc A -> hyp C -> type.
```

Related: Modality $\bullet A$ [Carbone, Montesi, CS '14]

Example

Cutting buyer and seller

Δ = u: hyp one,
p: proc (cut buyer λv. seller v) u,

After resolving the cut

Δ' = u : hyp one,
a : hyp (and string (imp nat one)),
p_1: proc buyer a,
p_2: proc (seller a) u

red/cut : proc (cut P (λv. Q v)) C $\multimap$
{ $\exists$a. proc P a $\otimes$
proc (Q a) C
}.

Implementing Admissibility of Cut

```
red/lolli : proc (lolliR (λu. P u)) C ⊸
            proc (lolliL Q1 (λv. Q2 v) C) C'' ⊸
            { ∃a. proc Q1 a ⊗
              ∃b. proc (P a) b ⊗
              proc (Q2 b) C''}.

red/one : proc end C ⊸
          proc (wait T C) C'' ⊸
          { proc T C''}.

red/and : proc (send M P) C ⊸
          proc (receive (λ!x. λu. Q !x u) C) C'' ⊸
          { ∃a. proc P a ⊗
            proc (Q !M a) C''}.
```

Demo

Session types -as- Judgments -as- Types

Theorem (Adequacy)

The representation in the Logical Framework is adequate, meaning that there exists a bijection between "processes" and objects.

Theorem (Reduction)

A "process" reduces to normal form iff the forward-chaining semantics reduces the encoded processes. The normal forms correspond.

Theorem (Concurrency)

Concurrent interleavings are truthfully represented in the framework.

Conclusion and Future Work

Conclusion and Future Work

- The logical framework community has developed tools useful for understanding session typing.
- Equational theory of the logical framework hides commutative cuts, when programming.
- We are currently working on extensions to multi-party session types. Preliminary results, see our papers in Concur '15 and '16.