

Behavioral Types and Logical Frameworks

An Introduction

Carsten Schürmann
IT University of Copenhagen

carsten@demtech.dk

March 7, 2019

Motivation

Buzzwords

- ▶ Concurrency
- ▶ Linear Logic
- ▶ Delegation
- ▶ Services
- ▶ Security

Motivating Example

On a laptop not far from here ...

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

Motivating Example

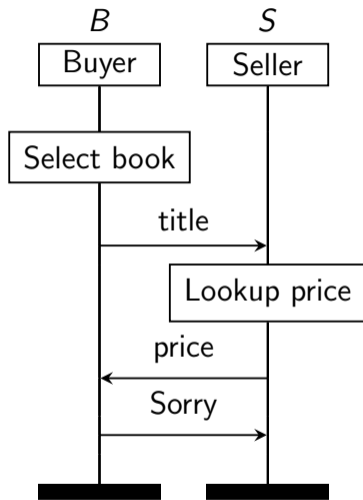
On a laptop not far from here ...

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

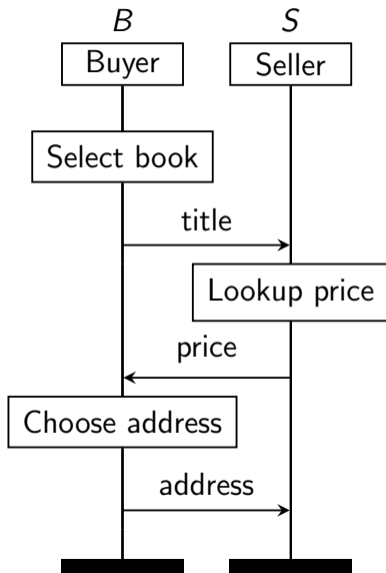
Observations:

- ▶ 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!



msc Yes!



Ingredients

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

Substructural Logical Frameworks

“ $25 + 5 = 3 \times 10$ ”



Substructural Logical Frameworks

“ $25 + 5 = 3 \times 10$ ”



Substructural Logical Frameworks

“ $25 + 5 = 3 \times 10$ ”



Substructural Logical Frameworks

“ $25 + 5 = 3 \times 10$ ”



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

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

R2 : $n \multimap n \multimap \{d\}$.

Substructural Logical Frameworks

“ $25 + 5 = 3 \times 10$ ”



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

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

R2 : $n \multimap n \multimap \{d\}$.

Observation

- ▶ 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

- ⊗ Connective to group facts
- Connective to express rewrite rules
- ∃ To create new evidence of facts
- ∏ 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 Substructural 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

Linear Logic – The Rules

$$\overline{\cdot \vdash 1} \text{ } 1R$$

$$\frac{\Delta \vdash C}{\Delta, 1 \vdash C} \text{ } 1L$$

$$\frac{\Delta_1 \vdash A \quad \Delta_2 \vdash B}{\Delta_1, \Delta_2 \vdash A \otimes B} \otimes R$$

$$\frac{\Delta, A, B \vdash C}{\Delta, A \otimes B \vdash C} \otimes L$$

$$\frac{\Delta, A \vdash B}{\Delta \vdash A \multimap B} \multimap R$$

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

$$\overline{A \vdash A} \text{ } \textit{init}$$

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

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, ...

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

Linear Logic - Primitive Types

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

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

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

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

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

Encoding in a Substructural Logical Framework

[Pfenning and Griffith '15]

Terms T

$1R$ end

$1L$ wait $[u]; T$

$\wedge R$ send $\langle M \rangle; T$

$\wedge L$ receive $(x) [u]; T$

$\supset R$ receive $(x); T$

$\supset L$ send $\langle M \rangle [u]; T$

$\otimes R, \otimes L$...

$\multimap R, \multimap L$...

Alternative: π -calculus

Related: Classical version of linear logic

[Caires & Pfenning '10, Wadler '12]

[Wadler'12]

Session Typing our Buyer Seller Example

$$\begin{array}{c} B \\ \vdash \text{string} \wedge (\text{nat} \supset 1) \end{array} \quad \begin{array}{c} S \\ (\text{string} \wedge (\text{nat} \supset 1)) \vdash 1 \end{array}$$

Comments

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

Buyer and Seller Example

$$\begin{array}{c} B \\ \vdash \text{string} \wedge (\text{nat} \supset 1) \end{array} \quad \begin{array}{c} S \\ u:(\text{string} \wedge (\text{nat} \supset 1)) \multimap 1 \vdash 1 \end{array}$$

Buyer B = `send <"Harry_potter">;`
`receive (price);`
`end`

Seller $S [u]$ = `receive (title) [u];`
`send <$45> [u];`
`wait [u];`
`end`

System C = `cut B (S [u]).`

Adding Choice

Linear Logic – The Additives

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

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

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

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

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

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

Linear Logic – The Additives

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

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

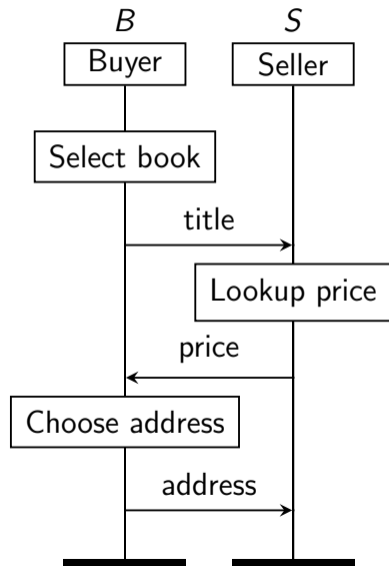
$$\frac{\Delta, u : A_2 \vdash T : C}{\Delta, u : A_1 \& A_2 \vdash \text{right} [u]; T : C} \&L_2$$

$$\frac{\Delta \vdash T : A_1}{\Delta \vdash \text{left}; T : A_1 \oplus A_2} \oplus R_1$$

$$\frac{\Delta \vdash T : A_2}{\Delta \vdash \text{right}; T : A_1 \oplus A_2} \oplus R_2$$

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

msc Yes!



```
B = send <"Harry_Potter">;  
receive (price);  
left;
```

```
send ("Berlin");  
end
```

```
S = receive (title) [u];  
send <$45> [u];  
offer[u](
```

```
left =>
```

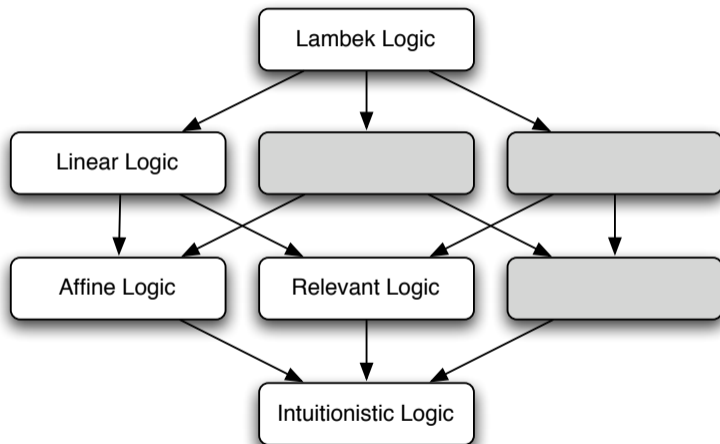
```
receive (address) [u];  
wait [u];  
end
```

```
right =>
```

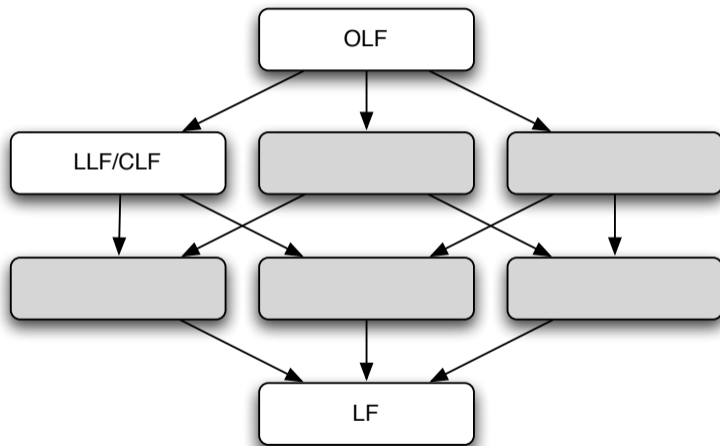
```
wait [u];  
end)
```

Substructural Logical Framework CLF

Substructural Logics



Substructural Logical Frameworks



Substructural Logics

$$\frac{A_1, \dots, A_m}{B_1, \dots, B_n} \textit{ name}$$

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

$$\textit{ name} : A_1 \otimes \dots \otimes A_m \multimap B_1 \otimes \dots \otimes B_n$$

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

$$\textit{ name} : A_1 \otimes \dots \otimes A_m \multimap \{B_1 \otimes \dots \otimes B_n\}$$

Execution as Proof Search

- ▶ Proof search

$$A$$
$$\vdots$$
$$B$$

corresponds to inhabitation of types.

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

- ▶ All terms are equal modulo interleavings
- ▶ No leftovers in the multi-set allowed
- ▶ Lollimon
- ▶ Focusing

[Lopez et al. '05]

[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:

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

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

CLF — Terms

Term syntax:

$$N ::= \lambda p. N \mid \langle N_1, N_2 \rangle \mid \langle \rangle \mid \{E\} \mid$$
$$c \mid x \mid N_1 N_2 \mid \pi_1 N \mid \pi_2 N$$
$$E ::= \text{let } \{p\} = N \text{ in } E \mid M$$
$$M ::= M_1 \otimes M_2 \mid 1 \mid N \mid !N \mid @N \mid [N, M]$$
$$p ::= p_1 \otimes p_2 \mid 1 \mid x \mid !x \mid @x \mid [x, p]$$

Objects

Expressions

Monadic objects

Patterns

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

$$\ulcorner A_1, \dots, A_n \overset{P}{\vdash} C \urcorner$$

=

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

Logical Framework Representation

<code>o</code> : type.	(Formulas)
<code>conc</code> : <code>o</code> -> type	(Conclusions)
<code>hyp</code> : <code>o</code> -> 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

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

Cut-Elimination

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

reduces to

Cut-Elimination

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

reduces to

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

Multi-Set Reduction

$$\left\{ \Delta, A \stackrel{P}{\vdash} B, \Delta_1 \stackrel{Q_1}{\vdash} A, \Delta_2, B \stackrel{Q_2}{\vdash} C \right\}$$

$$\Longrightarrow^*$$
$$\vdots$$
$$\Longrightarrow^*$$

$$\left\{ \cdot \stackrel{Q}{\vdash} 1 \right\}$$

Processes: always on, always connected!

$$\left\{ \Delta_1 \overset{Q_1}{\vdash} A, \Delta, u : A \overset{P}{\vdash} B \right\} \Longrightarrow \left\{ \Delta_1, \Delta \overset{R}{\vdash} B \right\}$$

Representation in CLF

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

Related: Modality $\bullet A$

[Carbone, Montesi, CS '14]

Example

Cutting buyer and seller

$$\Delta = u: \text{hyp one},$$
$$p: \text{proc } (\text{cut buyer } \lambda v. \text{ seller } v) u,$$

After resolving the cut

$$\Delta' = u : \text{hyp one},$$
$$a : \text{hyp } (\text{and string } (\text{imp nat one})),$$
$$p_1: \text{proc buyer } a,$$
$$p_2: \text{proc } (\text{seller } a) u$$
$$\text{red/cut} : \text{proc } (\text{cut } P (\lambda v. Q v)) C \multimap$$
$$\{ \exists a. \text{proc } P a \otimes$$
$$\text{proc } (Q a) C$$
$$\}.$$

Implementing Admissibility of Cut

```
red/lolli : proc (lolliR ( $\lambda u.$  P u)) C  $\multimap$   
  proc (lolliL Q1 ( $\lambda v.$  Q2 v) C) C''  $\multimap$   
  {  $\exists a.$  proc Q1 a  $\otimes$   
     $\exists b.$  proc (P a) b  $\otimes$   
    proc (Q2 b) C'' }.
```

```
red/one : proc end C  $\multimap$   
  proc (wait T C) C''  $\multimap$   
  { proc T C'' }.
```

```
red/and : proc (send M P) C  $\multimap$   
  proc (receive ( $\lambda !x.$   $\lambda u.$  Q !x u) C) C''  $\multimap$   
  {  $\exists a.$  proc P a  $\otimes$   
    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.