ILC: The Interactive Lambda Calculus

λ-Calculus for Universal Composibility or, *"Please, Halt Research with Interactive Turing Machines"*

Andrew Miller[1] and Matthew A. Hammer[2]

[1] University of Illinois
[2] University of Colorado Boulder

Abstract.

1 Introduction

2 Overview

3 ILC: Abridged Language Definition

4 ILC: Meta Theory

5 SaUCy Execution

Theorem 1 (Read determinism). *XXX*

A ILC: Full Language Definition

Done:

1. Define syntax for $A, C, \Delta, \Gamma, v, e, m$
2. Define judgement $m_1 \; ; m_2 \Rightarrow m_3$
3. Define judgement $m_1 \parallel m_2 \Rightarrow m_3$
4. Define judgement $\Delta; \Gamma \vdash e : C \rhd m$, except split, case, force

To do:

1. Define judgement $\Delta; \Gamma \vdash v : A$
2. Discuss typing for thunks
3. Define dynamic semantics judgement $e \longrightarrow e'$
4. State read determinism proof; prove it

$\texttt{execUC}(\mathcal{E}, \pi, \mathcal{A}, \mathcal{F})$

```
ν z2p z2f z2a p2f p2a a2f.
// The environment chooses SID, conf, and corrupted parties
let (Corrupted, SID, conf) = E{z2p, z2a, z2f}
// The protocol determines conf′
let conf′ = π.cmap(SID, conf)
| A{SID, conf, Corrupted, a2z, a2p, a2f}
| F{SID, conf′, Corrupted, f2z, f2p, f2a}
// Create instances of parties on demand
let partyMap = ref empty
let newPartyPID = do
  ν f2pp z2pp.
  @partyMap[PID].f2p := f2pp
  @partyMap[PID].z2p := z2pp
  | forever do {m ← pp2f; (PID, m) → f2p}
  | forever do {m ← pp2z; (PID, m) → z2p}
  | π{SID, conf, p2f/pp2z, p2z/pp2z}
let getParty PID =
    if PID ∉ partyMap then newParty PID
    return @partyMap[PID]
| forever do
  (PID, m) ← z2p
  if PID ∈ Corrupted then Z2P(PID, m) → p2a
  else m → (getParty PID).z2p
| forever do
  (PID, m) ← f2p
  if PID ∈ Corrupted then F2P(PID, m) → p2a
  else m → (getParty PID).f2p
| forever do
  | A2P2F(PID, m) ← a2p
    if PID ∈ Corrupted then (PID, m) → p2f
  | A2P2Z(PID, m) ← a2p
    if PID ∈ Corrupted then (PID, m) → p2z
```

Fig. 1. Definition of the SaUCy execution model. The environment, are run as concurrent processes. A new instance of the protocol π is created, on demand, for each party PID. Messages sent to honest parties are routed according to their PID; messages sent to corrupted parties are instead diverted to the adversary.

Value Types	A, B ::=	x	Value variable
		$\mid$ ()	Unit value
		$\mid$ nat	Natural number
		$\mid A \times B$	Product
		$\mid A + B$	Sum type
		$\mid \mathbf{!}A$	Intuitionistic type
		$\mid \mathbf{Rd}\, A$	Read channel
		$\mid \mathbf{Wr}\, A$	Write channel
		$\mid \mathbf{U}\, C$	Thunk type
Computation Types	C, D ::=	$A \to C$	Value-consuming computation
		$\mid \mathcal{F}_{\mathsf{A}}$	Value-producing computation
Linear Typing Contexts	Δ ::=	$\varepsilon \mid \Delta, x : A$	
Intuitionisitic Typing Contexts	Γ ::=	$\varepsilon \mid \Gamma, x : A$	

Fig. 2. Syntax of types and typing contexts

Values	v ::=	x	
		$\mid$ ()	Unit value
		$\mid n$	Natural number
		$\mid (v_1, v_2)$	Pair of values
		$\mid \mathtt{inj}_i\, v$	Injected value
		$\mid \mathtt{thunk}\, n$	Thunk (suspended, closed expression)
Expressions	e ::=	$\mathtt{split}(v, x_1.x_2.e)$	Pair elimination
		$\mid \mathtt{case}(v, x_1.e_1, x_2.e_2)$	Injection elimination
		$\mid \mathtt{ret}(v)$	Value-producing computation
		$\mid \mathtt{let}(e_1, x.e_2)$	Let-binding/sequencing
		$\mid e\ v$	Function application
		$\mid \lambda x.\, e$	Function abstraction
		$\mid \mathtt{force}(v)$	Unsuspend (force) a thunk
		$\mid \mathsf{wr}\, v_1 \leftarrow v_2$	Channel write
		$\mid \mathsf{rd}\, v$	Channel read
		$\mid \nu x.\, e$	Channel allocation
		$\mid e_1 \parallel e_2$	Parallel composition
		$\mid e_1 \,\&\, e_2$	Parallel choice

Fig. 3. Syntax of values and expressions

Modes m, n, p ::= $\mathsf{W} \mid \mathsf{R} \mid \mathsf{V}$ (Write, Read and Value)

$\boxed{m \parallel n \Rightarrow p}$ The parallel composition of modes m and n is mode p.

$$\frac{m \parallel n \Rightarrow p}{n \parallel m \Rightarrow p}\ \text{sym} \qquad \frac{}{\mathsf{W} \parallel \mathsf{V} \Rightarrow \mathsf{W}}\ \text{wv} \qquad \frac{}{\mathsf{W} \parallel \mathsf{R} \Rightarrow \mathsf{W}}\ \text{wr} \qquad \frac{}{\mathsf{R} \parallel \mathsf{R} \Rightarrow \mathsf{R}}\ \text{rr}$$

$\boxed{m\ ;\ n \Rightarrow p}$ The sequential composition of modes m and n is mode p.

$$\frac{}{\mathsf{V}\ ;\ n \Rightarrow n}\ \text{v}* \qquad \frac{}{\mathsf{W}\ ;\ \mathsf{V} \Rightarrow \mathsf{W}}\ \text{wv} \qquad \frac{}{\mathsf{R}\ ;\ n \Rightarrow \mathsf{R}}\ \text{r}*$$

$$\frac{}{\mathsf{W}\ ;\ \mathsf{R} \Rightarrow \mathsf{W}}\ \text{wr}$$

Note that in particular, the following mode compositions are *not derivable*:

- $\mathsf{W} \parallel \mathsf{W} \Rightarrow p$ is *not* derivable for any mode p
- $\mathsf{W}\ ;\ \mathsf{W} \Rightarrow p$ is *not* derivable for any mode p

Fig. 4. Syntax of modes; sequential and parallel mode composition.

$\boxed{\Delta; \Gamma \vdash e : C \rhd m}$ Under Δ and Γ, expression e has type C and mode m.

$$\frac{\Delta; \Gamma \vdash v : A}{\Delta; \Gamma \vdash \mathsf{ret}(v) : \mathcal{F}_{\mathsf{A}} \rhd \mathsf{V}}\ \text{ret} \qquad \frac{\begin{array}{c}\Delta_1; \Gamma \vdash e_1 : \mathcal{F}_{\mathsf{A}} \rhd m_1 \\ \Delta_2, x : A; \Gamma \vdash e_2 : C \rhd m_2 \\ m_1\ ;\ m_2 \Rightarrow m_3\end{array}}{\Delta_1, \Delta_2; \Gamma, x : A \vdash \mathsf{let}(e_1, x.e_2) : C \rhd m_3}\ \text{let}$$

$$\frac{\varepsilon; \Gamma \vdash v : A}{\varepsilon; \Gamma \vdash \mathsf{ret}(v) : \mathcal{F}_(\mathbf{!}A) \rhd \mathsf{V}}\ \text{ret!} \qquad \frac{\Delta_1; \Gamma \vdash v : \mathbf{!}A \qquad \Delta_2; \Gamma, x : A \vdash e : C \rhd m}{\Delta_1, \Delta_2; \Gamma, x : A \vdash \mathsf{let!}(v, x.e) : C \rhd m}\ \text{let!}$$

$$\frac{\Delta; \Gamma \vdash e : C \rhd m}{\Delta; \Gamma \vdash \lambda x.\, e : A \to C \rhd m}\ \text{lam} \qquad \frac{\Delta_1; \Gamma \vdash v : A \qquad \Delta_2; \Gamma \vdash e : A \to C \rhd m}{\Delta_1, \Delta_2; \Gamma \vdash e\, v : C \rhd m}\ \text{app}$$

$$\frac{\Delta, x : \big(\mathbf{Rd}\, A \times \mathbf{!}(\mathbf{Wr}\, A)\big); \Gamma \vdash e : C \rhd m}{\Delta; \Gamma \vdash \nu x.\ : C \rhd m}\ \text{nu}$$

$$\frac{\Delta; \Gamma \vdash v : \mathbf{Rd}\, A}{\Delta \vdash \mathsf{rd}\, v : \mathcal{F}_(A \times (\mathbf{Rd}\, A)) \rhd \mathsf{R}}\ \text{rd} \qquad \frac{\Delta_1; \Gamma \vdash v_1 : \mathbf{Rd}\, A \qquad \Delta_2; \Gamma \vdash v_2 : A}{\Delta_1, \Delta_2 \vdash \mathsf{wr}\, v_1 \leftarrow v_2 : \mathcal{F}_{\mathsf{unit}} \rhd \mathsf{W}}\ \text{wr}$$

$$\frac{\begin{array}{c}\Delta_1; \Gamma \vdash e_1 : C \rhd m_1 \\ \Delta_2; \Gamma \vdash e_2 : D \rhd m_2 \\ m_1 \parallel m_2 \Rightarrow m_3\end{array}}{\Delta_1, \Delta_2 \vdash e_1 \parallel e_2 : D \rhd m_3}\ \text{par1} \qquad \frac{\begin{array}{c}\Delta_1; \Gamma \vdash e_1 : C \rhd m_1 \\ \Delta_2; \Gamma \vdash e_2 : C \rhd m_2 \\ m_1 \parallel m_2 \Rightarrow m_3\end{array}}{\Delta_1, \Delta_2 \vdash e_1\ \&\ e_2 : C \rhd m_3}\ \text{par2}$$