

# **This is the Project Title**

*Your Name*

Master of Science  
Computer Science  
School of Informatics  
University of Edinburgh  
2020

# Abstract

Formal development of Frank.

# **Chapter 1**

## **Formalisation of Frank**

(data types)	$D$	(interfaces)	$I$
(value type variables)	$X$	(term variables)	$x, y, z, f$
(effect type variables)	$E$	(instance variables)	$s, a, b, c$
(value types)	$A, B ::= D \bar{R}$	(seeds)	$\sigma ::= \emptyset \mid E$
	$\mid \{C\} \mid X$	(abilities)	$\Sigma ::= \sigma \mid \Xi$
(computation types)	$C ::= \overline{T \rightarrow G}$	(extensions)	$\Xi ::= \mathfrak{t} \mid \Xi, I \bar{R}$
(argument types)	$T ::= \langle \Delta \rangle A$	(adaptors)	$\Theta ::= \mathfrak{t} \mid \Theta, I(S \rightarrow S')$
(return types)	$G ::= [\Sigma]A$	(adjustments)	$\Delta ::= \Theta \mid \Xi$
(type binders)	$Z ::= X \mid [E]$	(instance patterns)	$S ::= s \mid S a$
(type arguments)	$R ::= A \mid [\Sigma]$	(kind environments)	$\Phi, \Psi ::= \cdot \mid \Phi, Z$
(polytypes)	$P ::= \forall \bar{Z}. A$	(type environments)	$\Gamma ::= \cdot \mid \Gamma, x : A \mid \Gamma, f : P$
		(instance environments)	$\Omega ::= s : \Sigma \mid \Omega, a : I \bar{R}$

Figure 1.1: Types

(constructors)	$k$
(commands)	$c$
(uses)	$m ::= x \mid f \bar{R} \mid m \bar{n} \mid \uparrow(n : A)$
(constructions)	$n ::= \downarrow m \mid k \bar{n} \mid c \bar{R} \bar{n} \mid \{e\}$ $\mid \text{let } f : P = n \text{ in } n' \mid \text{letrec } \overline{f : P = e} \text{ in } n$ $\mid \langle \Theta \rangle n$
(computations)	$e ::= \overline{\bar{r} \mapsto n}$
(computation patterns)	$r ::= p \mid \langle c \bar{p} \rightarrow z \rangle \mid \langle x \rangle$
(value patterns)	$p ::= k \bar{p} \mid x$

Figure 1.2: Terms

$\Phi; \Gamma [\Sigma] \vdash m \Rightarrow A$	
$\frac{\text{T-VAR} \quad x : A \in \Gamma}{\Phi; \Gamma [\Sigma] \vdash x \Rightarrow A}$	$\frac{\text{T-POLYVAR} \quad \Phi \vdash \bar{R} \quad f : \forall \bar{Z}. A \in \Gamma}{\Phi; \Gamma [\Sigma] \vdash f \bar{R} \Rightarrow A[\bar{R}/\bar{Z}]}$
$\frac{\text{T-APP} \quad \Sigma' = \Sigma \quad (\Sigma \vdash \Delta_i \dashv \Sigma'_i)_i \quad \Phi; \Gamma [\Sigma] \vdash m \Rightarrow \{\langle \Delta \rangle A \rightarrow [\Sigma'] B\} \quad (\Phi; \Gamma [\Sigma'_i] \vdash n_i : A_i)_i}{\Phi; \Gamma [\Sigma] \vdash m \bar{n} \Rightarrow B}$	$\frac{\text{T-ASCRIBE} \quad \Phi; \Gamma [\Sigma] \vdash n : A}{\Phi; \Gamma [\Sigma] \vdash \uparrow(n : A) \Rightarrow A}$
$\Phi; \Gamma [\Sigma] \vdash n : A$	
$\frac{\text{T-SWITCH} \quad \Phi; \Gamma [\Sigma] \vdash m \Rightarrow A \quad A = B}{\Phi; \Gamma [\Sigma] \vdash \downarrow m : B}$	$\frac{\text{T-DATA} \quad k \bar{A} \in D \bar{R} \quad (\Phi; \Gamma [\Sigma] \vdash n_j : A_j)_j}{\Phi; \Gamma [\Sigma] \vdash k \bar{n} : D \bar{R}}$
$\frac{\text{T-COMMAND} \quad \Phi \vdash \bar{R} \quad c : \forall \bar{Z}. \bar{A} \rightarrow B \in \Sigma \quad (\Phi; \Gamma [\Sigma] \vdash n_j : A_j[\bar{R}/\bar{Z}])_j}{\Phi; \Gamma [\Sigma] \vdash c \bar{R} \bar{n} : B[\bar{R}/\bar{Z}]}$	$\frac{\text{T-THUNK} \quad \Phi; \Gamma \vdash e : C}{\Phi; \Gamma [\Sigma] \vdash \{e\} : \{C\}}$
$\frac{\text{T-LET} \quad P = \forall \bar{Z}. A \quad \Phi, \bar{Z}; \Gamma [\emptyset] \vdash n : A \quad \Phi; \Gamma, f : P [\Sigma] \vdash n' : B}{\Phi; \Gamma [\Sigma] \vdash \text{let } f : P = n \text{ in } n' : B}$	
$\frac{\text{T-LETREC} \quad (P_i = \forall \bar{Z}_i. \{C_i\})_i \quad (\Phi, \bar{Z}_i; \Gamma, \bar{f} : \bar{P} \vdash e_i : C)_i \quad \Phi; \Gamma, \bar{f} : \bar{P} [\Sigma] \vdash n : B}{\Phi; \Gamma [\Sigma] \vdash \text{letrec } \bar{f} : \bar{P} = e \text{ in } n : B}$	$\frac{\text{T-ADAPT} \quad \Sigma \vdash \Theta \dashv \Sigma' \quad \Phi; \Gamma [\Sigma'] \vdash n : A}{\Phi; \Gamma [\Sigma] \vdash \langle \Theta \rangle n : A}$
$\Phi; \Gamma \vdash e : C$	
$\frac{\text{T-COMP} \quad (\Phi \vdash r_{i,j} : T_j \dashv [\Sigma] \exists \Psi_{i,j}. \Gamma'_{i,j})_{i,j} \quad (\Phi, (\Psi_{i,j})_j; \Gamma, (\Gamma'_{i,j})_j [\Sigma] \vdash n_i : B)_i \quad ((r_{i,j})_i \text{ covers } T_j)_j}{\Phi; \Gamma \vdash ((r_{i,j})_j \mapsto n_i)_i : (T_j \rightarrow)_j [\Sigma] B}$	

Figure 1.3: Term Typing Rules

(uses)	$m ::= \dots \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil$
(constructions)	$n ::= \dots \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil$
(use values)	$u ::= x \mid f \bar{R} \mid \uparrow(v : A)$
(non-use values)	$v ::= k \bar{w} \mid \{e\}$
(construction values)	$w ::= \downarrow u \mid v$
(normal forms)	$t ::= w \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil$
(evaluation frames)	$\mathcal{F} ::= [] \bar{n} \mid u (\bar{t}, [], \bar{n}) \mid \uparrow([], A)$ $\mid \downarrow[] \mid k (\bar{w}, [], \bar{n}) \mid c \bar{R} (\bar{w}, [], \bar{n})$ $\mid \mathbf{let} f : P = [] \mathbf{in} n \mid \langle \Theta \rangle []$
(evaluation contexts)	$\mathcal{E} ::= [] \mid \mathcal{F}[\mathcal{E}]$

Figure 1.4: Runtime Syntax

$\Phi; \Gamma[\Sigma] \vdash m \Rightarrow A$	$\Phi; \Gamma[\Sigma] \vdash n : A$
$\frac{\text{T-FREEZE-USE} \quad \neg(\mathcal{E} \text{ handles } c) \quad \Phi; \Gamma[\Sigma] \vdash \mathcal{E}[c \bar{R} \bar{w}] \Rightarrow A}{\Phi; \Gamma[\Sigma] \vdash \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil \Rightarrow A}$	
$\frac{\text{T-FREEZE-CONS} \quad \neg(\mathcal{E} \text{ handles } c) \quad \Phi; \Gamma[\Sigma] \vdash \mathcal{E}[c \bar{R} \bar{w}] : A}{\Phi; \Gamma[\Sigma] \vdash \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil : A}$	

Figure 1.5: Frozen Commands

$$\begin{array}{c}
\boxed{m \rightsquigarrow_{\mathbf{u}} m'} \quad \boxed{n \rightsquigarrow_{\mathbf{c}} n'} \quad \boxed{m \longrightarrow_{\mathbf{u}} m'} \quad \boxed{n \longrightarrow_{\mathbf{c}} n'} \\
\\
\text{R-HANDLE} \\
\frac{k = \min_i \{i \mid \exists \bar{\theta}. (r_{i,j} : \langle \Delta_j \rangle A_j \leftarrow t_j \neg[\Sigma] \theta_j)_j\} \quad (r_{k,j} : \langle \Delta_j \rangle A_j \leftarrow t_j \neg[\Sigma] \theta_j)_j}{\uparrow(\{((r_{i,j})_j \rightarrow n_i)_i\} : \{\langle \Delta \rangle A \rightarrow [\Sigma] B\}) \bar{t} \rightsquigarrow_{\mathbf{u}} \uparrow((\bar{\theta}(n_k) : B))} \\
\\
\begin{array}{ccc}
\text{R-ASCRIBE-USE} & \text{R-ASCRIBE-CONS} & \text{R-LET} \\
\frac{}{\uparrow(\downarrow u : A) \rightsquigarrow_{\mathbf{u}} u} & \frac{}{\downarrow \uparrow(w : A) \rightsquigarrow_{\mathbf{c}} w} & \frac{}{\mathbf{let} \ f : P = w \ \mathbf{in} \ n \rightsquigarrow_{\mathbf{c}} n[\uparrow(w : P)/f]} \\
\\
\text{R-LETREC} & \frac{}{e = \bar{r} \rightarrow n} & \text{R-ADAPT} \\
\frac{}{\mathbf{letrec} \ f : P = e \ \mathbf{in} \ n' \rightsquigarrow_{\mathbf{c}} n'[\uparrow(\{\bar{r} \rightarrow \mathbf{letrec} \ f : P = e \ \mathbf{in} \ n\} : P)/f]} & & \frac{}{\langle \Theta \rangle w \rightsquigarrow_{\mathbf{c}} w} \\
\\
\text{R-FREEZE-COMM} \\
\frac{}{c \ \bar{R} \ \bar{w} \rightsquigarrow_{\mathbf{c}} [c \ \bar{R} \ \bar{w}]} \\
\\
\begin{array}{cc}
\text{R-FREEZE-FRAME-USE} & \text{R-FREEZE-FRAME-CONS} \\
\frac{\neg(\mathcal{F}[\mathcal{E}] \text{ handles } c)}{\mathcal{F}[[\mathcal{E}[c \ \bar{R} \ \bar{w}]]] \rightsquigarrow_{\mathbf{u}} [\mathcal{F}[\mathcal{E}[c \ \bar{R} \ \bar{w}]]]} & \frac{\neg(\mathcal{F}[\mathcal{E}] \text{ handles } c)}{\mathcal{F}[[\mathcal{E}[c \ \bar{R} \ \bar{w}]]] \rightsquigarrow_{\mathbf{c}} [\mathcal{F}[\mathcal{E}[c \ \bar{R} \ \bar{w}]]]} \\
\\
\begin{array}{cccc}
\text{R-LIFT-UU} & \text{R-LIFT-UC} & \text{R-LIFT-CU} & \text{R-LIFT-CC} \\
\frac{m \rightsquigarrow_{\mathbf{u}} m'}{\mathcal{E}[m] \longrightarrow_{\mathbf{u}} \mathcal{E}[m']} & \frac{m \rightsquigarrow_{\mathbf{u}} m'}{\mathcal{E}[m] \longrightarrow_{\mathbf{c}} \mathcal{E}[m']} & \frac{n \rightsquigarrow_{\mathbf{c}} n'}{\mathcal{E}[n] \longrightarrow_{\mathbf{u}} \mathcal{E}[n']} & \frac{n \rightsquigarrow_{\mathbf{c}} n'}{\mathcal{E}[n] \longrightarrow_{\mathbf{c}} \mathcal{E}[n']}
\end{array}
\end{array}
\end{array}$$

Figure 1.6: Operational Semantics

$$\boxed{r : T \leftarrow t \neg[\Sigma] \theta}$$

$$\begin{array}{c} \text{B-VALUE} \\ \Sigma \vdash \Delta \dashv \Sigma' \\ p : A \leftarrow w \dashv \theta \\ \hline p : \langle \Delta \rangle A \leftarrow w \neg[\Sigma] \theta \end{array}$$

B-REQUEST

$$\begin{array}{c} \Sigma \vdash \Delta \dashv \Sigma' \quad \mathcal{E} \text{ poised for } c \\ \Delta = \Theta \mid \Xi \quad c : \forall \bar{Z}. \bar{B} \rightarrow B' \in \Xi \quad (p_i : B_i \leftarrow w_i \dashv \theta_i)_i \\ \hline \langle c \bar{p} \rightarrow z \rangle : \langle \Delta \rangle A \leftarrow [\mathcal{E}[c \bar{R} \bar{w}]] \neg[\Sigma] \bar{\theta}[\uparrow(\{x \mapsto \mathcal{E}[x]\} : \{B' \rightarrow [\Sigma']A\})/z] \end{array}$$

B-CATCHALL-VALUE

$$\begin{array}{c} \Sigma \vdash \Delta \dashv \Sigma' \\ \hline \langle x \rangle : \langle \Delta \rangle A \leftarrow w \neg[\Sigma] [\uparrow(\{w\} : \{[\Sigma']A\})/x] \end{array}$$

B-CATCHALL-REQUEST

$$\begin{array}{c} \Sigma \vdash \Delta \dashv \Sigma' \quad \mathcal{E} \text{ poised for } c \\ \Delta = \Theta \mid \Xi \quad c : \forall \bar{Z}. \bar{B} \rightarrow B' \in \Xi \\ \hline \langle x \rangle : \langle \Delta \rangle A \leftarrow [\mathcal{E}[c \bar{R} \bar{w}]] \neg[\Sigma] [\uparrow(\{[\mathcal{E}[c \bar{R} \bar{w}]]\} : \{[\Sigma']A\})/x] \end{array}$$

$$\boxed{p : A \leftarrow w \dashv \theta}$$

B-VAR

$$\frac{}{x : A \leftarrow w \dashv [\uparrow(w : A)/x]}$$

B-DATA

$$\frac{k \bar{A} \in D \bar{R} \quad (p_i : A_i \leftarrow w_i \dashv \theta_i)_i}{k \bar{p} : D \bar{R} \leftarrow k \bar{w} \dashv \bar{\theta}}$$

Figure 1.7: Pattern Binding



# Chapter 2

## Arbitrary Thread Interruption

### 2.1 Relaxing Catches

$$\begin{array}{c}
 \text{B-CATCHALL-REQUEST-LOOSE} \\
 \Sigma \vdash \Delta \dashv \Sigma' \\
 \hline
 \langle x \rangle : \langle \Delta \rangle A \leftarrow \lceil \mathcal{E}[c \ \bar{R} \ \bar{w}] \rceil \text{-}[\Sigma] \ [\uparrow(\{\lceil \mathcal{E}[c \ \bar{R} \ \bar{w}] \rceil\} : \{\lceil \Sigma' \rceil A\}) / x]
 \end{array}$$

Figure 2.1: Updated B-CATCHALL-REQUEST

### 2.2 Interrupting Arbitrary Terms

(uses)	$m ::= \dots \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil$
(constructions)	$n ::= \dots \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil$
(use values)	$u ::= x \mid f \bar{R} \mid \uparrow(v : A)$
(non-use values)	$v ::= k \bar{w} \mid \{e\}$
(construction values)	$w ::= \downarrow u \mid v$
(normal forms)	$t ::= w \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil \mid !(m)$
(evaluation frames)	$\mathcal{F} ::= [] \bar{n} \mid u (\bar{t}, [], \bar{n}) \mid \uparrow([], : A)$ $\mid \downarrow[] \mid k (\bar{w}, [], \bar{n}) \mid c \bar{R} (\bar{w}, [], \bar{n})$ $\mid \mathbf{let} f : P = [] \mathbf{in} n \mid \langle \Theta \rangle []$
(evaluation contexts)	$\mathcal{E} ::= [] \mid \mathcal{F}[\mathcal{E}]$

Figure 2.2: Runtime Syntax, Updated with Suspension of Uses

$$\begin{array}{c}
\text{B-CATCHALL-INTERRUPT} \\
\hline
\Sigma \vdash \Delta \dashv \Sigma' \\
\hline
\langle x \rangle : \langle \Delta \rangle A \leftarrow !(m) \dashv [\Sigma] [\uparrow(\{m\} : \{\Sigma' \} A)] / x
\end{array}$$

Figure 2.3: Catching Interrupts rule.

## 2.3 Freezing

Another way of doing it is to just let any use become frozen, in the same way as commands become frozen once invoked.

**TODO:** Do we need the 'hoisting' rules for general suspended terms?

## 2.4 Yielding

Note that R-YIELD-EF relies on the predicate  $\mathcal{F}[\mathcal{E}]$  allows  $c$ . This states that the Yield interface is in the ability of the term in the evaluation context.

For any frame apart from argument frames,  $\mathcal{F}[\mathcal{E}]$  allows  $c = \text{false}$ . In this case, it is defined as follows;

$$\uparrow(v : \{\overline{\langle \Delta \rangle A \rightarrow [\Sigma] B}\}) (\bar{t}, [], \bar{n}) \text{ allows } c = \begin{cases} \Xi \text{ allows } c & \text{if } |\bar{n}| = 0 \text{ where } \Sigma = \sigma \mid \Xi \\ \text{false} & \text{otherwise} \end{cases}$$

**TODO:** What happens for 0-ary constructors? I don't know if it's exactly how I think it is.

R-INTERRUPT

$$\frac{}{m \rightsquigarrow_u!(m)}$$

Figure 2.4: Use interruption rule

$m \rightsquigarrow_u m'$	$n \rightsquigarrow_c n'$	$m \longrightarrow_u m'$	$n \longrightarrow_c n'$
R-FREEZE-USE		R-FREEZE-COMM	
$\frac{}{m \rightsquigarrow_u [m]}$		$\frac{}{c \bar{R} \bar{w} \rightsquigarrow_c [c \bar{R} \bar{w}]}$	
R-FREEZE-FRAME-USE		R-FREEZE-FRAME-CONS	
$\frac{}{\mathcal{F}[[m]] \rightsquigarrow_u [\mathcal{F}[m]]}$		$\frac{}{\mathcal{F}[[m]] \rightsquigarrow_c [\mathcal{F}[m]]}$	
R-FREEZE-FRAME-USE		R-FREEZE-FRAME-CONS	
$\frac{\neg(\mathcal{F}[\mathcal{E}] \text{ handles } c)}{\mathcal{F}[[\mathcal{E}[c \bar{R} \bar{w}]]] \rightsquigarrow_u [\mathcal{F}[\mathcal{E}[c \bar{R} \bar{w}]]]}$		$\frac{\neg(\mathcal{F}[\mathcal{E}] \text{ handles } c)}{\mathcal{F}[[\mathcal{E}[c \bar{R} \bar{w}]]] \rightsquigarrow_c [\mathcal{F}[\mathcal{E}[c \bar{R} \bar{w}]]]}$	

Figure 2.5: Updated Freezing

For an extension  $\Xi$ , the allows predicate is defined as

$$\begin{aligned} \mathbf{1} \text{ allows } c &= \text{false} \\ (\Xi, I \bar{R}) \text{ allows } c &= \begin{cases} \text{true} & \text{if } c \in I \\ \Xi \text{ allows } c & \text{otherwise} \end{cases} \end{aligned}$$

Informally,  $\mathcal{F}[\mathcal{E}] \text{ allows } c$  is true when  $\mathcal{F}[\mathcal{E}]$  is a handler with the command  $c$  as a member of an interface in its ability, *and* all of the arguments have been evaluated (this is what the  $|\bar{n}| = 0$  constraint expresses).

We also make use of an auxiliary combinator  $;\dots$ . This is the traditional sequential composition operator, where both arguments are evaluated and the result of the second one is returned. In the context of R-YIELD-EF this means we will perform the yield effect and then the use  $m$ , but discard the result from yield.

(uses)	$m ::= \dots \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil \mid \boxed{m}$
(constructions)	$n ::= \dots \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil \mid \boxed{m}$
(use values)	$u ::= x \mid f \bar{R} \mid \uparrow(v : A)$
(non-use values)	$v ::= k \bar{w} \mid \{e\}$
(construction values)	$w ::= \downarrow u \mid v$
(normal forms)	$t ::= w \mid \lceil \mathcal{E}[c \bar{R} \bar{w}] \rceil \mid \boxed{m}$
(evaluation frames)	$\mathcal{F} ::= [] \bar{n} \mid u (\bar{t}, [], \bar{n}) \mid \uparrow([], : A)$ $\mid \downarrow[] \mid k (\bar{w}, [], \bar{n}) \mid c \bar{R} (\bar{w}, [], \bar{n})$ $\mid \text{let } f : P = [] \text{ in } n \mid \langle \Theta \rangle []$
(evaluation contexts)	$\mathcal{E} ::= [] \mid \mathcal{F}[\mathcal{E}]$

Figure 2.6: Runtime Syntax, Updated with Freezing of Uses

$$\begin{array}{c}
\text{B-CATCHALL-INTERRUPT} \\
\hline
\Sigma \vdash \Delta \dashv \Sigma' \\
\hline
\langle x \rangle : \langle \Delta \rangle A \leftarrow \boxed{m} \dashv [\Sigma] [\uparrow(\{m\} : \{\Sigma' \} A)] / x
\end{array}$$

Figure 2.7: Catching Frozen Terms rule.

## 2.5 Counting

In practise we count up through the amount of R-HANDLE rules we apply and only insert the yield when this count exceeds a threshold value  $t_y$ .

So we supplement the operational semantics with a *counter*  $c_y$ , so that our transitions are e.g.  $m; c_y \rightsquigarrow_u m'; c_y'$ . We adopt the convention that, when the counter is not mentioned in a transition<sup>1</sup>, the counter stays the same. Hence e.g.  $m \rightsquigarrow_u m'$  desugars to  $m; c_y \rightsquigarrow_u m'; c_y$ .

So to get our counting semantics we just need to supplement Figure [?] with the updated rule in

## 2.6 Counting — Take Two

In the final semantics we count up through the number of R-HANDLE uses. This lets us track when to next insert a yield.

<sup>1</sup>That is, the transition is of the form  $m \rightsquigarrow_u m'$ .

$$\boxed{m \rightsquigarrow_u m'}$$

R-YIELD

$$\frac{\Sigma = \sigma \mid \Xi \quad \text{Yield} \in \Xi}{\uparrow(\{n\} : \{\Sigma\}A) \rightsquigarrow_u \uparrow(\text{let } f_n : \{\text{Unit} \rightarrow [\Sigma]A\} = \{- \mapsto n\} \text{ in } \text{let } y : \{\Sigma\}\text{Unit} = \{\text{yield}!\} \text{ in } \{f_n(\downarrow(y!))\} : \{\Sigma\}A))}$$

$$\frac{\text{R-YIELD-EF} \quad \mathcal{F}[\mathcal{E}] \text{ allows Yield}}{\mathcal{F}[\mathcal{E}[m]] \rightsquigarrow_u \mathcal{F}[\mathcal{E}[\text{yield}!; m] ]}$$

Figure 2.8: Inserting Yields

**TODO:** In R-Yield-EF we have to let terms inside eval CTXs be also uses. Does this not mean that we can then put any term in either one? Is that a problem?

$$\boxed{m \rightsquigarrow_u m'} \quad \boxed{n \rightsquigarrow_c n'} \quad \boxed{m \longrightarrow_u m'} \quad \boxed{n \longrightarrow_c n'}$$

R-HANDLE

$$\frac{k = \min_i \{i \mid \exists \bar{\theta}. (r_{i,j} : \langle \Delta_j \rangle A_j \leftarrow t_j \dashv [\Sigma] \theta_j)_j\} \quad (r_{k,j} : \langle \Delta_j \rangle A_j \leftarrow t_j \dashv [\Sigma] \theta_j)_j}{\uparrow(\{((r_{i,j})_j \rightarrow n_i)_i\} : \{\langle \Delta \rangle A \rightarrow [\Sigma] B\}) \bar{t}; c_y \rightsquigarrow_u \uparrow((\bar{\theta}(n_k) : B)); c_y + 1}$$

Figure 2.9: R-Handle with counting.

We use a counter for this, labelled  $c_y$  in the semantics. This counter essentially has two states; it is either simply counting, i.e. is  $c(n)$  for some  $n$  or is a message to yield as soon as possible, i.e. `yield`.

To increment this counter, we use a slightly modified version of addition, denoted  $+_c$ . This is simply defined as

$$x +_c y = \begin{cases} c(x + y) & \text{if } x + y \leq t_y \\ \text{yield} & \text{otherwise} \end{cases}$$

where  $t_y$  is the threshold at which we force a yield. Thus the updated semantics for R-HANDLE follows.

$$\boxed{m \rightsquigarrow_u m'}$$

$$\text{R-YIELD}$$

$$\frac{c_y \geq t_y}{m; c_y \rightsquigarrow_u \text{let yield : [Yield]Unit} = \text{yin } m; 0}$$

Figure 2.10: Inserting Yields, when over counter

$$\boxed{m \rightsquigarrow_u m'} \quad \boxed{n \rightsquigarrow_c n'} \quad \boxed{m \longrightarrow_u m'} \quad \boxed{n \longrightarrow_c n'}$$

$$\text{R-HANDLE-COUNT}$$

$$\frac{k = \min_i \{i \mid \exists \bar{\theta}. (r_{i,j} : \langle \Delta_j \rangle A_j \leftarrow t_j - [\Sigma] \theta_j)_j\} \quad (r_{k,j} : \langle \Delta_j \rangle A_j \leftarrow t_j - [\Sigma] \theta_j)_j}{\uparrow(\{((r_{i,j})_j \rightarrow n_i)_i\} : \{\overline{(\Delta)A} \rightarrow [\Sigma]B\}) \bar{t}; \text{c}(n) \rightsquigarrow_u \uparrow((\bar{\theta}(n_k) : B)); n +_c 1}$$

Figure 2.11: R-Handle with better counting.

**TODO:** Talk about how everything else implicitly just passes by; only if the counter is in the form  $c(n)$ .

R-HANDLE-COUNT in Figure ?? expresses that when handling, if our counter is of the form  $c(n)$  — i.e. we do not have to yield — we perform the handling step as usual and increment the counter, potentially yielding if we need to. R-YIELD-CAN in Figure 2.13 expresses that if the counter is in the form `yield` and the evaluation context allows us to yield then we *must* yield, and reset the counter to 0. R-YIELD-CAN'T says that if the evaluation context does not permit yielding, but  $m$  would otherwise reduce to some term  $m'$ , then we perform that reduction inside the context. This is important; we do not block the rest of the code from reducing if we need to yield. The programmer may of course never want to yield; they may want their system to remain fully synchronous. As such they can simply never write the `Yield` interface in their program anywhere and the program will not yield.

The rest of the operational semantics remains the same. We adopt the syntactic sugar that any non-labelled transition  $m \rightsquigarrow_u m'$  (resp.  $\rightsquigarrow_c$ ) is shorthand for  $m; c(n) \rightsquigarrow_u m'; c(n)$ ; that is, it only applies when the counter is plain and non-blocking, and it leaves the counter unmodified.

$$\boxed{m \rightsquigarrow_u m'}$$

$$\begin{array}{c}
\text{R-YIELD-CAN} \\
\mathcal{F}[\mathcal{E}] \text{ allows Yield} \\
\hline
\mathcal{F}[\mathcal{E}[m]]; \text{yield} \rightsquigarrow_u \mathcal{F}[\mathcal{E}[\text{yield!}; m]]; c(0)
\end{array}$$

$$\begin{array}{c}
\text{R-YIELD-CAN'T} \\
\neg(\mathcal{F}[\mathcal{E}] \text{ allows Yield}) \quad m; c(n) \rightsquigarrow_u m'; c' \\
\hline
\mathcal{F}[\mathcal{E}[m]]; \text{yield} \rightsquigarrow_u \mathcal{F}[\mathcal{E}[m]]; \text{yield}
\end{array}$$

Figure 2.12: Inserting Yields when forced to.

## 2.7 Soundness

$$\boxed{m \rightsquigarrow_u m'}$$

$$\begin{array}{c}
\text{R-YIELD-CAN} \\
\mathcal{F} \text{ allows Yield} \\
\hline
\mathcal{F}[n]; \text{yield} \rightsquigarrow_u \mathcal{F}[\text{yield!}; n]; c(0)
\end{array}$$

$$\begin{array}{c}
\text{R-YIELD-CAN'T} \\
\neg(\mathcal{F} \text{ allows Yield}) \quad n; c(k) \rightsquigarrow_u n'; c' \\
\hline
\mathcal{F}[n]; \text{yield} \rightsquigarrow_u \mathcal{F}[n']; \text{yield}
\end{array}$$

Figure 2.13: Inserting Yields when forced to; with newer version

We now state the soundness property for our extended system, as well as the subject reduction theorem needed for this proof. Our system is nothing more than the system of [?] with extra rules; as such we omit most of the details.

**Theorem 1 (Subject Reduction)** • If  $\Phi; \Gamma[\Sigma] \vdash m \Rightarrow A$  and  $m; c_y \rightsquigarrow_u m'; c'_y$  then  $\Phi; \Gamma[\Sigma] \vdash m' \Rightarrow A$ .

• If  $\Phi; \Gamma[\Sigma] \vdash n : A$  and  $n; c_y \rightsquigarrow_c n'; c'_y$  then  $\Phi; \Gamma[\Sigma] \vdash n' : A$ .

By induction on the transitions  $\rightsquigarrow_u, \rightsquigarrow_c$ .

We first consider the two possible states for  $c_y$ . If it is in the form  $c(n)$ , then the reduction rules are simply the same as in [? ], as we do not change the counter. The only exception to this is the updated R-HANDLE rule, which is essentially the same except for modifications to the counter; regardless of the counter, the resulting term  $m'$  still remains the same type.

Thus the only new cases are R-YIELD-CAN and R-YIELD-CAN'T.

**Case R-YIELD-CAN**

By the assumption we have that  $\mathcal{F}$  allows yield. This only holds if the evaluation frame is of the form

$$\uparrow(v : \{\overline{\langle \Delta \rangle A \rightarrow [\Sigma] B}\}) (\bar{t}, [], \bar{n}')$$

.

Filling the hole then, we must have that

$$\mathcal{F}[n] = \uparrow(v : \{\overline{\langle \Delta \rangle A \rightarrow [\Sigma] B}\}) (\bar{t}, n, \bar{n}')$$

$$\mathcal{F}[\text{yield!}; n] = \uparrow(v : \{\overline{\langle \Delta \rangle A \rightarrow [\Sigma] B}\}) (\bar{t}, \text{yield!}; n, \bar{n}')$$

Then assume  $\Phi; \Gamma[\Sigma] \vdash \mathcal{F}[n] : B$ . By inversion on T-APP, we have  $\Phi; \Gamma[\Sigma'] \vdash n : A_{|\bar{t}|}$ .

**TODO:** Do we need to worry about the adaptors?

**TODO:** Fill in space here — need to be sure that the ability permits what we think it does.

Clearly then  $\Phi; \Gamma[\Sigma'] \vdash \text{yield!}; n : A_{|\bar{t}|}$ , as we know from  $\mathcal{F}$  allows yield that  $\in \Sigma'_{|\bar{t}|}$ .

**Case R-YIELD-CAN'T** This case is more straightforward. By the assumption we have that the evaluation frame  $\mathcal{F}$  does not permit yielding, but the term inside the frame could otherwise reduce.

Assume  $\Phi; \Gamma[\Sigma] \vdash \mathcal{F}[n] : A$ , and therefore  $\Phi; \Gamma[\Sigma] \vdash n : A'$ . By the assumption and subject reduction,  $\Phi; \Gamma[\Sigma] \vdash n' : A'$ . Then clearly  $\Phi; \Gamma[\Sigma] \vdash \mathcal{F}[n'] : A$ .

**Theorem 2 (Type Soundness)** • If  $\cdot; \cdot[\Sigma] \vdash m \Rightarrow A$  then either  $m$  is a normal form such that  $m$  respects  $\Sigma$  or there exists a unique  $\cdot; \cdot[\Sigma] \vdash m' \Rightarrow A$  such that  $m \longrightarrow_u m'$ .

- If  $\cdot; \cdot[\Sigma] \vdash n : A$  then either  $n$  is a normal form such that  $n$  respects  $\Sigma$  or there exists a unique  $\cdot; \cdot[\Sigma] \vdash n' : A$  such that  $n \longrightarrow_c n'$ .

The proof proceeds by simultaneous induction on  $\cdot; \cdot[\Sigma] \vdash m \Rightarrow A$  and  $\cdot; \cdot[\Sigma] \vdash n : A$ , with use of Theorem 1.



## **Chapter 3**

### **Use-Only Reductions**

Here we talk about how to change the operational semantics so that only uses may reduce.

(uses)	$m ::= \dots \mid [\mathcal{E}[c \bar{R} \bar{w}]]$
(constructions)	$n ::= \dots$
(use values)	$u ::= x \mid f \bar{R} \mid \uparrow(v : A)$
(non-use values)	$v ::= k \bar{w} \mid \{e\}$
(construction values)	$w ::= \downarrow u \mid v$
(normal forms)	$t ::= w \mid [\mathcal{E}[c \bar{R} \bar{w}]]$
(evaluation frames)	$\mathcal{F} ::= [] \bar{n} \mid u (\bar{t}, [], \bar{n}) \mid \uparrow([], : A)$ $\mid \downarrow[] \mid k (\bar{w}, [], \bar{n}) \mid c \bar{R} (\bar{w}, [], \bar{n})$ $\mid \mathbf{let} f : P = [] \mathbf{in} n \mid \langle \Theta \rangle []$
(evaluation contexts)	$\mathcal{E} ::= [] \mid \mathcal{F}[\mathcal{E}]$

Figure 3.1: Runtime Syntax for Use-Only Reductions

**TODO:** Remove command uses from constructions?

**TODO:** Change eval ctxs (i can't remember what to change to)?

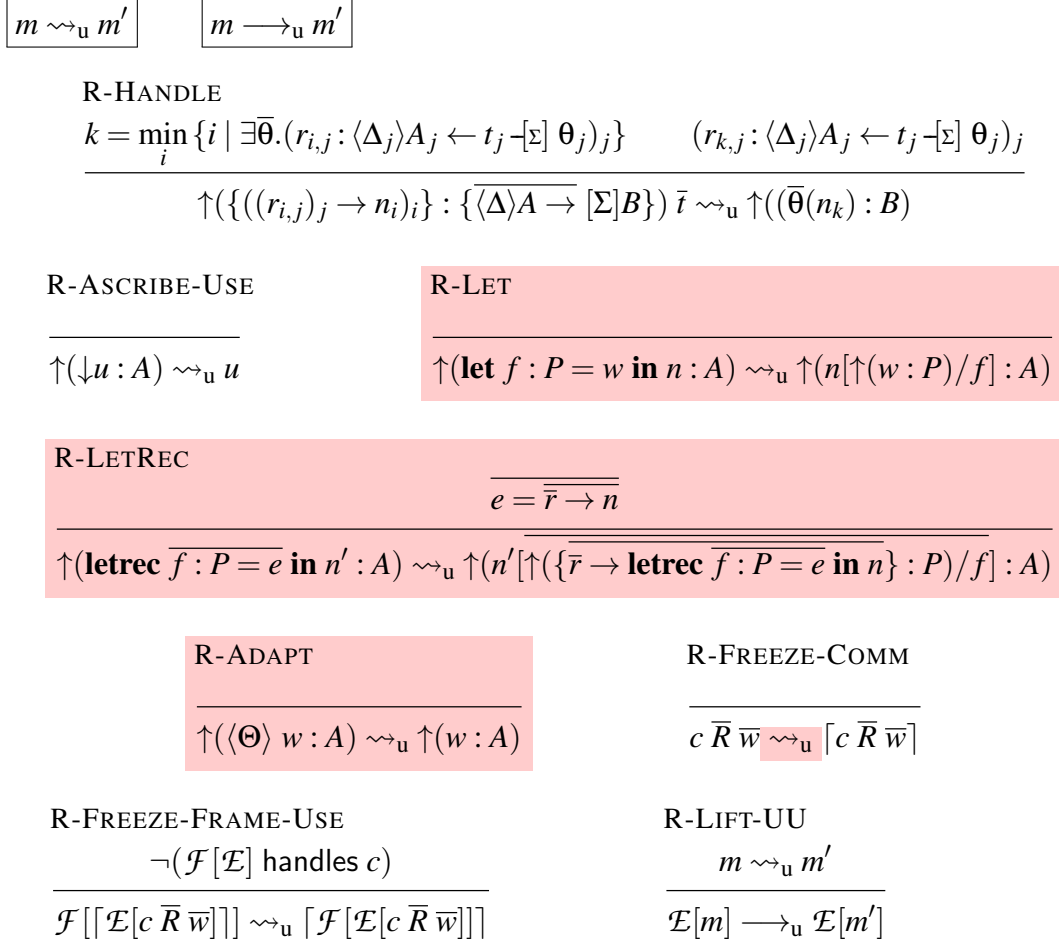


Figure 3.2: Operational Semantics

**TODO:** Can R-Ascribe-Cons just get removed?

**TODO:** Can we just move commands to be in uses?

# **Appendix A**

## **Remaining Formalisms**

$$\boxed{\Sigma \vdash \Delta \dashv \Sigma'}$$

$$\begin{array}{c} \text{A-ADJ} \\ \hline \Sigma \vdash \Theta \dashv \Sigma' \quad \Sigma' \vdash \Xi \dashv \Sigma'' \\ \hline \Sigma \vdash \Theta | \Xi \dashv \Sigma'' \end{array}$$

$$\boxed{\Sigma \vdash \Xi \dashv \Sigma'}$$

$$\begin{array}{c} \text{A-EXT-ID} \\ \hline \Sigma \vdash \mathbf{1} \dashv \Sigma \end{array}$$

$$\begin{array}{c} \text{A-EXT-SNOC} \\ \hline \Sigma \vdash \Xi \dashv \Sigma' \\ \hline \Sigma \vdash \Xi, I \bar{R} \dashv \Sigma', I \bar{R} \end{array}$$

$$\boxed{\Sigma \vdash \Theta \dashv \Sigma'}$$

$$\begin{array}{c} \text{A-ADAPT-ID} \\ \hline \Sigma \vdash \mathbf{1} \dashv \Sigma \end{array}$$

$$\begin{array}{c} \text{A-ADAPT-SNOC} \\ \hline \Sigma \vdash \Theta \dashv \Sigma' \quad \Sigma' \vdash I(S \rightarrow S') \dashv \Sigma'' \\ \hline \Sigma \vdash \Theta, I(S \rightarrow S') \dashv \Sigma'' \end{array}$$

$$\boxed{\Sigma \vdash I(S \rightarrow S') \dashv \Sigma'}$$

$$\begin{array}{c} \text{A-ADAPT-COM} \\ \hline \Sigma \vdash S : I \dashv \Sigma'; \Omega \quad \Omega \vdash S' : I \dashv \Xi \quad \Sigma' \vdash \Xi \dashv \Sigma'' \\ \hline \Sigma \vdash I(S \rightarrow S') \dashv \Sigma'' \end{array}$$

$$\boxed{\Sigma \vdash S : I \dashv \Sigma'; \Omega}$$

$$\text{I-PAT-ID}$$

$$\Sigma \vdash s : I \dashv \Sigma; s : \Sigma$$

$$\text{I-PAT-BIND}$$

$$\Sigma \vdash S : I \dashv \Sigma'; \Omega$$

$$\Sigma, I \bar{R} \vdash S a : I \dashv \Sigma'; \Omega, a : I \bar{R}$$

$$\text{I-PAT-SKIP}$$

$$\Sigma \vdash S a : I \dashv \Sigma'; \Omega \quad I \neq I'$$

$$\Sigma, I' \bar{R} \vdash S a : I \dashv \Sigma', I' \bar{R}; \Omega$$

$$\boxed{\Omega \vdash S : I \dashv \Xi}$$

$$\begin{array}{c} \text{I-INST-ID} \\ \hline s \in \text{dom}(\Omega) \\ \hline \Omega \vdash s : I \dashv \mathbf{1} \end{array}$$

$$\begin{array}{c} \text{I-INST-LKP} \\ \hline a \in \text{dom}(\Omega) \quad \Omega \vdash S : I \dashv \Xi \quad \Omega(a) = I \bar{R} \\ \hline \Omega \vdash S a : I \dashv \Xi, I \bar{R} \end{array}$$

Figure A.1: Action of an Adjustment on an Ability and Auxiliary Judgements

$$\mathcal{X} ::= A \mid C \mid T \mid G \mid Z \mid R \mid P \mid \sigma \mid \Sigma \mid \Xi \mid \Theta \mid \Delta \mid \Gamma \mid \exists \Psi. \Gamma \mid \Omega$$

$\Phi \vdash \mathcal{X}$

$\frac{}{\Phi, X \vdash X}$ <p>WF-VAL</p>	$\frac{}{\Phi, [E] \vdash E}$ <p>WF-EFF</p>	$\frac{\Phi, \bar{Z} \vdash A}{\Phi \vdash \forall \bar{Z}. A}$ <p>WF-POLY</p>
$\frac{(\Phi \vdash R)_i}{\Phi \vdash D \bar{R}}$ <p>WF-DATA</p>	$\frac{\Phi \vdash C}{\Phi \vdash \{C\}}$ <p>WF-THUNK</p>	$\frac{(\Phi \vdash T)_i \quad \Phi \vdash G}{\Phi \vdash \bar{T} \rightarrow G}$ <p>WF-COMP</p>
$\frac{\Phi \vdash \Delta \quad \Phi \vdash A}{\Phi \vdash \langle \Delta \rangle A}$ <p>WF-ARG</p>		
$\frac{\Phi \vdash \Sigma \quad \Phi \vdash A}{\Phi \vdash [\Sigma] A}$ <p>WF-RET</p>	$\frac{\Phi \vdash \Sigma}{\Phi \vdash [\Sigma]}$ <p>WF-ABILITY</p>	$\frac{}{\Phi \vdash \emptyset}$ <p>WF-PURE</p>
$\frac{}{\Phi \vdash \mathbf{1}}$ <p>WF-ID</p>		$\frac{\Phi \vdash \Xi \quad (\Phi \vdash R)_i}{\Phi \vdash \Xi, I \bar{R}}$ <p>WF-EXT</p>
$\frac{\Phi \vdash \Theta}{\Phi \vdash \Theta, I (S \rightarrow S')}$ <p>WF-ADAPT</p>		
$\frac{}{\Phi \vdash \cdot}$ <p>WF-EMPTY</p>	$\frac{\Phi \vdash \Gamma \quad \Phi \vdash A}{\Phi \vdash \Gamma, x : A}$ <p>WF-MONO</p>	$\frac{\Phi \vdash \Gamma \quad \Phi \vdash P}{\Phi \vdash \Gamma, f : P}$ <p>WF-POLY</p>
$\frac{\Phi, \Psi \vdash \Gamma}{\Phi \vdash \exists \Psi. \Gamma}$ <p>WF-EXISTENTIAL</p>		$\frac{\Phi \vdash \Omega \quad (\Phi \vdash R)_i}{\Phi \vdash \Omega, x : I \bar{R}}$ <p>WF-INTERFACE</p>

Figure A.2: Well-Formedness Rules

$$\boxed{\Phi \vdash p : A \dashv \Gamma}$$

$$\begin{array}{c}
\text{P-VAR} \\
\hline
\Phi \vdash x : A \dashv x : A
\end{array}
\qquad
\begin{array}{c}
\text{P-DATA} \\
k \bar{A} \in D \bar{R} \quad (\Phi \vdash p_i : A_i \dashv \Gamma)_i \\
\hline
\Phi \vdash k \bar{p} : D \bar{R} \dashv \bar{\Gamma}
\end{array}$$

$$\boxed{\Phi \vdash r : T \dashv [\Sigma] \exists \Psi. \Gamma}$$

$$\begin{array}{c}
\text{P-VALUE} \\
\Sigma \vdash \Delta \dashv \Sigma' \quad \Phi \vdash p : A \dashv \Gamma \\
\hline
\Phi \vdash p : \langle \Delta \rangle A \dashv [\Sigma] \Gamma
\end{array}
\qquad
\begin{array}{c}
\text{P-CATCHALL} \\
\Sigma \vdash \Delta \dashv \Sigma' \\
\hline
\Phi \vdash \langle x \rangle : \langle \Delta \rangle A \dashv [\Sigma] x : \{[\Sigma'] A\}
\end{array}$$

$$\begin{array}{c}
\text{P-COMMAND} \\
\Sigma \vdash \Delta \dashv \Sigma' \quad \Delta = \Theta \mid \Xi \quad c : \forall \bar{Z}. \bar{A} \rightarrow \bar{B} \in \Xi \quad (\Phi, \bar{Z} \vdash p_i : A_i \dashv \Gamma_i)_i \\
\hline
\Phi \vdash \langle c \bar{p} \rightarrow z \rangle : \langle \Delta \rangle B' \dashv [\Sigma] \exists \bar{Z}. \bar{\Gamma}, z : \{ \langle \mathbf{1} \mid \mathbf{1} \rangle B \rightarrow [\Sigma'] B' \}
\end{array}$$

Figure A.3: Pattern Matching Typing Rules