

# Compiling from $F_i^+$ to JavaScript

Yaozhu Sun

December 11, 2023

## Syntax of $F_i^+$

Types	$A, B, C ::= \top \mid \perp \mid \mathbb{Z} \mid X \mid A \rightarrow B \mid \forall X * A. B \mid \{\ell : A\} \mid A \& B$
Expressions	$e ::= \{\} \mid n \mid x \mid \mathbf{fix} \, x : A. e \mid \lambda x : A. e : B \mid e_1 \, e_2 \mid \Lambda X * A. e : B \mid e \, A$ $\mid \{\ell = e\} \mid e.\ell \mid e_1, e_2 \mid e : A$
Type indices	$T ::= \mathbb{Z} \mid \vec{T} \mid T^\forall \mid \{\ell : T\} \mid T_1 \& T_2$
JavaScript code	$J ::= \emptyset \mid J_1; J_2 \mid \mathbf{code}$

$$\boxed{\Gamma \vdash e \Leftrightarrow A \rightsquigarrow J \mid z^\pm}$$

(Type-directed compilation)

$\frac{\text{J-GEN} \quad \Gamma \vdash e \Leftrightarrow A \rightsquigarrow J \mid z^-}{\Gamma \vdash e \Leftrightarrow A \rightsquigarrow \mathbf{code} \mid z^+}$	$\frac{\text{J-TOP}}{\Gamma \vdash \{\} \Rightarrow \top \rightsquigarrow \emptyset \mid z^-}$	$\frac{\text{J-TOPABS} \quad \lceil B \rceil}{\Gamma \vdash \lambda x : A. e : B \Rightarrow A \rightarrow B \rightsquigarrow \emptyset \mid z^-}$
$\frac{\text{J-TOPTABS} \quad \lceil B \rceil}{\Gamma \vdash \Lambda X * A. e : B \Rightarrow \forall X * A. B \rightsquigarrow \emptyset \mid z^-}$	$\frac{\text{J-TOPRCD} \quad \Gamma \vdash e \Rightarrow A \quad \lceil A \rceil}{\Gamma \vdash \{\ell = e\} \Rightarrow \{\ell : A\} \rightsquigarrow \emptyset \mid z^-}$	
$\frac{\text{J-INT} \quad T =  \mathbb{Z} }{\Gamma \vdash n \Rightarrow \mathbb{Z} \rightsquigarrow \mathbf{code} \mid z^-}$	$\frac{\text{J-VAR} \quad x : A \in \Gamma}{\Gamma \vdash x \Rightarrow A \rightsquigarrow \mathbf{code} \mid z^-}$	$\frac{\text{J-VARGEN} \quad x : A \in \Gamma}{\Gamma \vdash x \Rightarrow A \rightsquigarrow \emptyset \mid z^+}$
$\frac{\text{J-FIX} \quad \Gamma, x : A \vdash e \Leftarrow A \rightsquigarrow J \mid z^-}{\Gamma \vdash \mathbf{fix} \, x : A. e \Rightarrow A \rightsquigarrow \mathbf{code} \mid z^-}$	$\frac{\text{J-ABS} \quad T = \overrightarrow{ B } \quad \Gamma, x : A \vdash e \Leftarrow B \rightsquigarrow J \mid y^-}{\Gamma \vdash \lambda x : A. e : B \Rightarrow A \rightarrow B \rightsquigarrow \mathbf{code} \mid z^-}$	
$\frac{\text{J-APP} \quad \begin{array}{l} \Gamma \vdash e_1 \Rightarrow A \rightsquigarrow J_1 \mid x^+ \\ \Gamma \vdash e_2 \Rightarrow B \rightsquigarrow J_2 \mid \mathit{this}^- \\ \Gamma \vdash x : A \bullet y_0 : B \rightsquigarrow J_3 \mid z : C \\ T_s = \mathbf{itoa} \mid B \end{array}}{\Gamma \vdash e_1 \, e_2 \Rightarrow C \rightsquigarrow \mathbf{code} \mid z^-}$	$\frac{\text{J-TABS} \quad T =  B ^\forall \quad \Gamma, X * A \vdash e \Leftarrow B \rightsquigarrow J \mid y^-}{\Gamma \vdash \Lambda X * A. e : B \Rightarrow \forall X * A. B \rightsquigarrow \mathbf{code} \mid z^-}$	
$\frac{\text{J-TAPP} \quad \begin{array}{l} \Gamma \vdash e \Rightarrow B \rightsquigarrow J_1 \mid y^+ \\ \Gamma \vdash y : B \bullet A \rightsquigarrow J_2 \mid z : C \end{array}}{\Gamma \vdash e \, A \Rightarrow C \rightsquigarrow J_1; J_2 \mid z^-}$	$\frac{\text{J-RCD} \quad T = \{\ell :  A \} \quad \Gamma \vdash e \Rightarrow A \rightsquigarrow J \mid y^+}{\Gamma \vdash \{\ell = e\} \Rightarrow \{\ell : A\} \rightsquigarrow \mathbf{code} \mid z^-}$	$\frac{\text{J-PROJ} \quad \begin{array}{l} \Gamma \vdash e \Rightarrow A \rightsquigarrow J_1 \mid y^+ \\ y : A \bullet \{\ell\} \rightsquigarrow J_2 \mid z : B \end{array}}{\Gamma \vdash e.\ell \Rightarrow B \rightsquigarrow J_1; J_2 \mid z^-}$

$\text{J-MERGE} \quad \frac{\Gamma \vdash e_1 \Rightarrow A \rightsquigarrow J_1 \mid z^- \quad \Gamma \vdash e_2 \Rightarrow B \rightsquigarrow J_2 \mid z^-}{\Gamma \vdash e_1, e_2 \Rightarrow A \& B \rightsquigarrow J_1; J_2 \mid z^-}$			$\text{J-ANNO} \quad \frac{\Gamma \vdash e \Leftarrow A \rightsquigarrow J \mid z^\pm}{\Gamma \vdash e : A \Rightarrow A \rightsquigarrow J \mid z^\pm}$		
$\text{J-DEF} \quad \frac{\Gamma \vdash e_1 \Rightarrow A \rightsquigarrow J_1 \mid x^- \quad \Gamma, x : A \vdash e_2 \Rightarrow B \rightsquigarrow J_2 \mid z^-}{\Gamma \vdash x = e_1; e_2 \Rightarrow B \rightsquigarrow \text{code} \mid z^-}$		$\text{J-SUB} \quad \frac{\Gamma \vdash e \Rightarrow A \rightsquigarrow J_1 \mid x^+ \quad x : A <:^+ y : B \rightsquigarrow J_2}{\Gamma \vdash e \Leftarrow B \rightsquigarrow J_1; J_2 \mid y^-}$		$\text{J-SUBEQUIV} \quad \frac{A \doteq B \quad \Gamma \vdash e \Rightarrow A \rightsquigarrow J \mid z^\pm}{\Gamma \vdash e \Leftarrow B \rightsquigarrow J \mid z^\pm}$	
<pre>/* J-Gen */ var z = {}; J;  /* J-Int */ z[T] = n;  /* J-Var */ copy(z, x);  /* J-Fix */ var x = z; J;  /* J-Abs */ z[T] = (x, y) =&gt; { J };</pre>		<pre>/* J-App */ J1; var y0 = {}; for (const T of Ts)   y0.__defineGetter__(T,     () =&gt; {       for (const T of Ts)         delete this[T];       J2;       return this[T];     }); J3;</pre>		<pre>/* J-TAbs */ z[T] = (X, y) =&gt; { J };  /* J-Rcd */ z.__defineGetter__(T, () =&gt; {   J;   delete this[T];   return this[T] = y; });  /* J-Def */ export var x = {}; J1; J2;</pre>	

**Copying properties** n.b. there seems to be some alternatives:

- `Object.assign(z, x)` does not properly copy getters;
- `Object.defineProperties(z, Object.getOwnPropertyDescriptors(x))` is proper but slow;
- `Object.setPrototypeOf(z, x)` does the prototype trick but is even slower.

```
function copy(z, x) {
  for (const prop in x) {
    var getter = x.__lookupGetter__(prop);
    if (getter) z.__defineGetter__(prop, getter);
    else z[prop] = x[prop];
  }
}
```

$$\boxed{\Gamma \vdash x : A \bullet p \rightsquigarrow J \mid z : B}$$

(Distributive application)

$$\frac{\text{A-Top} \quad \lceil A \rceil}{\Gamma \vdash x : A \bullet p \rightsquigarrow \emptyset \mid z : \top}$$

$$\frac{\text{A-ARROW} \quad y : C <: ^+ \text{ this} : A \rightsquigarrow J \quad T = \overrightarrow{|B|} \quad Ts = \mathbf{itoa} \mid A \mid}{\Gamma \vdash x : A \rightarrow B \bullet y : C \rightsquigarrow \mathbf{code} \mid z : B}$$

$$\frac{\text{A-ARROWEQUIV} \quad A \doteq C \quad T = \overrightarrow{|B|}}{\Gamma \vdash x : A \rightarrow B \bullet y : C \rightsquigarrow \mathbf{code} \mid z : B}$$

$$\frac{\text{A-ALL} \quad \Gamma \vdash A * C \quad T = |B|^\forall \quad Ts = \mathbf{itoa} \mid C \mid}{\Gamma \vdash x : \forall X * A. B \bullet C \rightsquigarrow \mathbf{code} \mid z : B[X \mapsto C]}$$

$$\frac{\text{A-AND} \quad \Gamma \vdash x : A \bullet p \rightsquigarrow J_1 \mid z : A' \quad \Gamma \vdash x : B \bullet p \rightsquigarrow J_2 \mid z : B'}{\Gamma \vdash x : A \& B \bullet p \rightsquigarrow J_1; J_2 \mid z : A' \& B'}$$

```
/* A-Arrow */
var y0 = {};
for (const T of Ts)
  y0.__defineGetter__(T, () => {
    for (const T of Ts)
      delete this[T];
    J;
    return this[T];
  });
x[T](y0, z);
```

```
/* A-ArrowEquiv */
x[T](y, z);

/* A-All */
x[T](Ts, z);
```

$$\boxed{x : A \bullet \{\ell\} \rightsquigarrow J \mid z : B}$$

(Distributive projection)

$$\frac{\text{P-Top} \quad \lceil A \rceil}{x : A \bullet \{\ell\} \rightsquigarrow \emptyset \mid z : \top}$$

$$\frac{\text{P-RcdEq} \quad T = \{\ell : |A|\}}{x : \{\ell : A\} \bullet \{\ell\} \rightsquigarrow \mathbf{code} \mid z : A}$$

$$\frac{\text{P-RcdNEq} \quad \ell_1 \neq \ell_2 \quad T = \{\ell : |A|\}}{x : \{\ell_1 : A\} \bullet \{\ell_2\} \rightsquigarrow \emptyset \mid z : \top}$$

$$\frac{\text{P-AND} \quad x : A \bullet \{\ell\} \rightsquigarrow J_1 \mid z : A' \quad x : B \bullet \{\ell\} \rightsquigarrow J_2 \mid z : B'}{x : A \& B \bullet \{\ell\} \rightsquigarrow J_1; J_2 \mid z : A' \& B'}$$

```
/* P-RcdEq */
copy(z, x[T]);
```

$$\boxed{x : A <:\pm y : B \rightsquigarrow J}$$

(Coercive subtyping)

$$\frac{\text{S-EQUIV} \quad A \doteq B}{x : A <:\text{+} y : B \rightsquigarrow \text{code}}$$

$$\frac{\text{S-TOP} \quad \lceil B \rceil}{x : A <:\pm y : B \rightsquigarrow \emptyset}$$

$$\frac{\text{S-BOT} \quad T = |A|}{x : \perp <:\pm y : A \rightsquigarrow \text{code}}$$

$$\frac{\text{S-SPLIT} \quad \begin{array}{l} B_1 \triangleleft B \triangleright B_2 \\ y_1 : B_1 \triangleright z : B \triangleleft y_2 : B_2 \rightsquigarrow J_3 \\ x : A <:\pm y_1 : B_1 \rightsquigarrow J_1 \\ x : A <:\pm y_2 : B_2 \rightsquigarrow J_2 \end{array}}{x : A <:\pm z : B \rightsquigarrow \text{code}}$$

$$\frac{\text{S-INT} \quad T = |\mathbb{Z}|}{x : \mathbb{Z} <:\pm y : \mathbb{Z} \rightsquigarrow \text{code}}$$

$$\frac{\text{S-VAR}}{x : X <:\pm y : X \rightsquigarrow \text{code}}$$

$$\frac{\text{S-ARROW} \quad \begin{array}{l} T_1 = \overrightarrow{|A_2|} \quad T_2 = \overrightarrow{|B_2|} \\ Ts = \mathbf{itoa} \mid A_1 \mid \\ x_1 : B_1 <:\text{+} \text{this} : A_1 \rightsquigarrow J_1 \\ x_2 : A_2 <:\text{+} y_2 : B_2 \rightsquigarrow J_2 \end{array}}{x : A_1 \rightarrow A_2 <:\pm y : B_1 \rightarrow B_2 \rightsquigarrow \text{code}}$$

$$\frac{\text{S-ALL} \quad \begin{array}{l} T_1 = |A_2|^\forall \\ T_2 = |B_2|^\forall \quad B_1 <: A_1 \\ x_0 : A_2 <:\text{+} y_0 : B_2 \rightsquigarrow J \end{array}}{x : \forall X * A_1. A_2 <:\pm y : \forall X * B_1. B_2 \rightsquigarrow \text{code}}$$

$$\frac{\text{S-RCD} \quad \begin{array}{l} T_1 = \{\ell : |A|\} \\ T_2 = \{\ell : |B|\} \\ x_0 : A <:\text{+} y_0 : B \rightsquigarrow J \end{array}}{x : \{\ell : A\} <:\pm y : \{\ell : B\} \rightsquigarrow \text{code}}$$

$$\frac{\text{S-ANDL} \quad x : A <:\text{-} y : C \rightsquigarrow J}{x : A \& B <:\pm y : C \rightsquigarrow J}$$

$$\frac{\text{S-ANDR} \quad x : B <:\text{-} y : C \rightsquigarrow J}{x : A \& B <:\pm y : C \rightsquigarrow J}$$

```
/* S-Equiv */
copy(y, x);
```

```
/* S-Bot */
y[T] = null;
```

```
/* S-Split */
var y1 = {}; // if y1 != z
var y2 = {}; // if y2 != z
J1; J2; J3;
```

```
/* S-Int */
y[T] = x[T];
```

```
/* S-Var */
copy(y, x);
```

```
/* S-Arrow */
y[T2] = (x1, y2) => {
  var y1 = {};
  for (const T of Ts)
    y1.__defineGetter__(T,
      () => {
        for (const T of Ts)
          delete this[T];
        J1;
        return this[T];
      });
  var x2 = {};
  x[T1](y1, x2);
  J2;
};
```

```
/* S-All */
y[T2] = (X, y0) => {
  var x0 = {};
  x[T1](X, x0);
  J;
};

/* S-Rcd */
y.__defineGetter__(T2, () => {
  var x0 = x[T1];
  var y0 = {}; J;
  delete this[T];
  return this[T] = y0;
});
```

$$\boxed{x : A \triangleright z : C \triangleleft y : B \rightsquigarrow J}$$

(Coercive merging)

$$\begin{array}{c} \text{M-AND} \\ \hline z : A \triangleright z : A \& B \triangleleft z : B \rightsquigarrow \emptyset \end{array} \quad \begin{array}{c} \text{M-ARROW} \\ \hline \begin{array}{c} T = \overrightarrow{|B|} \\ T_1 = \overrightarrow{|B_1|} \quad T_2 = \overrightarrow{|B_2|} \\ y_1 : B_1 \triangleright y : B \triangleleft y_2 : B_2 \rightsquigarrow J \end{array} \\ \hline x_1 : A \rightarrow B_1 \triangleright z : A \rightarrow B \triangleleft x_2 : A \rightarrow B_2 \rightsquigarrow \text{code} \end{array}$$

$$\begin{array}{c} \text{M-ALL} \\ \hline \begin{array}{c} T = |B|^\forall \\ T_1 = |B_1|^\forall \quad T_2 = |B_2|^\forall \\ y_1 : B_1 \triangleright y : B \triangleleft y_2 : B_2 \rightsquigarrow J \end{array} \\ \hline x_1 : \forall X * A. B_1 \triangleright z : \forall X * A. B \triangleleft x_2 : \forall X * A. B_2 \rightsquigarrow \text{code} \end{array}$$

$$\begin{array}{c} \text{M-RCD} \\ \hline \begin{array}{c} T = \{\ell : |A|\} \\ T_1 = \{\ell : |A_1|\} \\ T_2 = \{\ell : |A_2|\} \\ y_1 : A_1 \triangleright y : A \triangleleft y_2 : A_2 \rightsquigarrow J \end{array} \\ \hline x_1 : \{\ell : A_1\} \triangleright z : \{\ell : A\} \triangleleft x_2 : \{\ell : A_2\} \rightsquigarrow \text{code} \end{array}$$

```

/* M-Arrow */
z[T] = (p, y) => {
  var y1 = {}; // if y1 != y
  var y2 = {}; // if y2 != y
  x1[T1](p, y1);
  x2[T2](p, y2);
  J;
};

/* M-All */
z[T] = (X, y) => {
  var y1 = {}; // if y1 != y
  var y2 = {}; // if y2 != y
  x1[T1](X, y1);
  x2[T2](X, y2);
  J;
};

/* M-Rcd */
z.__defineGetter__(T, () => {
  var y = {};
  var y1 = {}; // if y1 != y
  var y2 = {}; // if y2 != y
  copy(y1, x1[T1]);
  copy(y2, x2[T2]);
  J;
  delete this[T];
  return this[T] = y;
});

```