Compiling from F_i^+ to JavaScript

Yaozhu Sun

January 26, 2024

Syntax of F_i^+

Types	$A, B, C ::= \top \mid \bot \mid \mathbb{Z} \mid X \mid A \to B \mid \forall X{*}A.\ B \mid \{\ell : A\} \mid A \,\&\, B$
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	$T ::= \mathbb{Z} \mid \overrightarrow{T} \mid T^{\forall} \mid \{\ell : T\} \mid T_1 \,\&\, T_2$
JavaScript code	$J ::= \varnothing \mid J_1; J_2 \mid \texttt{code}$

$\boxed{\Gamma \vdash e \Leftrightarrow A \rightsquigarrow J \mid z^{\pm}}$ *(Type-directed compilation)*

J-Gen
$$\frac{\Gamma \vdash e \Leftrightarrow A \rightsquigarrow J \mid z^-}{\Gamma \vdash e \Leftrightarrow A \rightsquigarrow \texttt{code} \mid z^+}$$

J-Top
$$\frac{}{\Gamma \vdash \{\} \Rightarrow \top \rightsquigarrow \varnothing \mid z^-}$$

J-TopAbs
$$\frac{\rceil B \lceil}{\Gamma \vdash \lambda x{:}A.\ e{:}B \Rightarrow A \to B \rightsquigarrow \varnothing \mid z^-}$$

J-TopTAbs
$$\frac{\rceil B \lceil}{\Gamma \vdash \Lambda X{*}A.\ e{:}B \Rightarrow \forall X{*}A.\ B \rightsquigarrow \varnothing \mid z^-}$$

J-TopRcd
$$\frac{\Gamma \vdash e \Rightarrow A \qquad \rceil A \lceil}{\Gamma \vdash \{\ell = e\} \Rightarrow \{\ell : A\} \rightsquigarrow \varnothing \mid z^-}$$

J-Int
$$\frac{T = |\mathbb{Z}|}{\Gamma \vdash n \Rightarrow \mathbb{Z} \rightsquigarrow \texttt{code} \mid z^-}$$

J-Var
$$\frac{x : A \in \Gamma}{\Gamma \vdash x \Rightarrow A \rightsquigarrow \texttt{code} \mid z^-}$$

J-VarGen
$$\frac{x : A \in \Gamma}{\Gamma \vdash x \Rightarrow A \rightsquigarrow \varnothing \mid x^+}$$

J-Fix
$$\frac{\Gamma, x : A \vdash e \Leftarrow A \rightsquigarrow J \mid z^-}{\Gamma \vdash \mathbf{fix}\, x{:}A.\ e \Rightarrow A \rightsquigarrow \texttt{code} \mid z^-}$$

J-Abs
$$\frac{T = |\overrightarrow{B}| \qquad \Gamma, x : A \vdash e \Leftarrow B \rightsquigarrow J \mid y^-}{\Gamma \vdash \lambda x{:}A.\ e{:}B \Rightarrow A \to B \rightsquigarrow \texttt{code} \mid z^-}$$

J-App
$$\frac{\Gamma \vdash e_1 \Rightarrow A \rightsquigarrow J_1 \mid x^+ \qquad \Gamma \vdash e_2 \Rightarrow B \rightsquigarrow J_2 \mid y^+ \qquad \Gamma \vdash x : A \bullet y : B \rightsquigarrow J_3 \mid z : C}{\Gamma \vdash e_1\ e_2 \Rightarrow C \rightsquigarrow J_1; J_2; J_3 \mid z^-}$$

J-TAbs
$$\frac{T = |B|^{\forall} \qquad \Gamma, X * A \vdash e \Leftarrow B \rightsquigarrow J \mid y^-}{\Gamma \vdash \Lambda X{*}A.\ e{:}B \Rightarrow \forall X{*}A.\ B \rightsquigarrow \texttt{code} \mid z^-}$$

J-TApp
$$\frac{\Gamma \vdash e \Rightarrow B \rightsquigarrow J_1 \mid y^+ \qquad \Gamma \vdash y : B \bullet A \rightsquigarrow J_2 \mid z : C}{\Gamma \vdash e\ A \Rightarrow C \rightsquigarrow J_1; J_2 \mid z^-}$$

J-Rcd
$$\frac{T = \{\ell : |A|\} \qquad \Gamma \vdash e \Rightarrow A \rightsquigarrow J \mid y^+}{\Gamma \vdash \{\ell = e\} \Rightarrow \{\ell : A\} \rightsquigarrow \texttt{code} \mid z^-}$$

J-Proj
$$\frac{\Gamma \vdash e \Rightarrow A \rightsquigarrow J_1 \mid y^+ \qquad y : A \bullet \{\ell\} \rightsquigarrow J_2 \mid z : B}{\Gamma \vdash e.\ell \Rightarrow B \rightsquigarrow J_1; J_2 \mid z^-}$$

J-Merge

$$\frac{\Gamma \vdash e_1 \Rightarrow A \rightsquigarrow J_1 \mid z^- \quad \Gamma \vdash e_2 \Rightarrow B \rightsquigarrow J_2 \mid z^- \quad \Gamma \vdash A * B}{\Gamma \vdash e_1 \,,\, e_2 \Rightarrow A \,\&\, B \rightsquigarrow J_1; J_2 \mid z^-}$$

J-Anno

$$\frac{\Gamma \vdash e \Leftarrow A \rightsquigarrow J \mid z^{\pm}}{\Gamma \vdash e : A \Rightarrow A \rightsquigarrow J \mid z^{\pm}}$$

J-Def

$$\frac{\Gamma \vdash e_1 \Rightarrow A \rightsquigarrow J_1 \mid x^- \quad \Gamma, x : A \vdash e_2 \Rightarrow B \rightsquigarrow J_2 \mid z^-}{\Gamma \vdash x = e1; e_2 \Rightarrow B \rightsquigarrow \texttt{code} \mid z^-}$$

J-Sub

$$\frac{\Gamma \vdash e \Rightarrow A \rightsquigarrow J_1 \mid x^+ \quad x : A <:^+ y : B \rightsquigarrow J_2}{\Gamma \vdash e \Leftarrow B \rightsquigarrow J_1; J_2 \mid y^-}$$

J-SubEquiv

$$\frac{A \eqsim B \quad \Gamma \vdash e \Rightarrow A \rightsquigarrow J \mid z^{\pm}}{\Gamma \vdash e \Leftarrow B \rightsquigarrow J \mid z^{\pm}}$$

```
/* 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) => { J };

/* J-TAbs */
z[T] = (X, y) => { J };
```

```
/* J-Rcd */
z.__defineGetter__(T, () => {
  J;
  delete this[T];
  return this[T] = y;
});

/* J-Def */
export var x = {};
J1;
J2;
```

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)*

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

A-Arrow
$$\frac{y : C <:^{+} y_0 : A \rightsquigarrow J \qquad T = |\overrightarrow{B}|}{\Gamma \vdash x : A \to B \bullet y : C \rightsquigarrow \texttt{code} \mid z : B}$$

A-ArrowEquiv
$$\frac{A \coloneqq C \qquad T = |\overrightarrow{B}|}{\Gamma \vdash x : A \to B \bullet y : C \rightsquigarrow \texttt{code} \mid z : B}$$

A-All
$$\frac{\Gamma \vdash A * C \qquad T = |B|^{\forall} \qquad Ts = \textbf{itoa} \mid C \mid}{\Gamma \vdash x : \forall X * A.\ B \bullet C \rightsquigarrow \texttt{code} \mid z : B[X \mapsto C]}$$

A-And
$$\frac{\Gamma \vdash x : A \bullet p \rightsquigarrow J_1 \mid z : A' \qquad \Gamma \vdash x : B \bullet p \rightsquigarrow J_2 \mid z : B'}{\Gamma \vdash x : A \,\&\, B \bullet p \rightsquigarrow J_1; J_2 \mid z : A' \,\&\, B'}$$

```
/* A-Arrow */
var y0 = {};
J;
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)*

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

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

P-RcdNeq
$$\frac{\ell_1 \neq \ell_2 \qquad T = \{\ell : |A|\}}{x : \{\ell_1 : A\} \bullet \{\ell_2\} \rightsquigarrow \varnothing \mid z : \top}$$

P-And
$$\frac{x : A \bullet \{\ell\} \rightsquigarrow J_1 \mid z : A' \qquad x : B \bullet \{\ell\} \rightsquigarrow J_2 \mid z : B'}{x : A \,\&\, B \bullet \{\ell\} \rightsquigarrow J_1; J_2 \mid z : A' \,\&\, B'}$$

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

$\boxed{x : A <:^{\pm} \ y : B \rightsquigarrow J}$ *(Coercive subtyping)*

S-Equiv
$$\frac{A \coloneqq B}{x : A <:^{+} \ y : B \rightsquigarrow \texttt{code}}$$

S-Top
$$\frac{\rceil B \lceil}{x : A <:^{\pm} \ y : B \rightsquigarrow \varnothing}$$

S-Bot
$$\frac{T = |A|}{x : \bot <:^{\pm} \ y : A \rightsquigarrow \texttt{code}}$$

S-Split
$$\frac{\begin{array}{c} B_1 \lhd B \rhd B_2 \\ y_1 : B_1 \rhd z : B \lhd y_2 : B_2 \rightsquigarrow J_3 \\ x : A <:^{\pm} \ y_1 : B_1 \rightsquigarrow J_1 \\ x : A <:^{\pm} \ y_2 : B_2 \rightsquigarrow J_2 \end{array}}{x : A <:^{\pm} \ z : B \rightsquigarrow \texttt{code}}$$

S-Int
$$\frac{T = |\mathbb{Z}|}{x : \mathbb{Z} <:^{\pm} \ y : \mathbb{Z} \rightsquigarrow \texttt{code}}$$

S-Var
$$\frac{}{x : X <:^{\pm} \ y : X \rightsquigarrow \texttt{code}}$$

S-Arrow
$$\frac{\begin{array}{c} T_1 = |\overrightarrow{A_2}| \qquad T_2 = |\overrightarrow{B_2}| \\ Ts = \textbf{itoa} \mid A_1 \mid \\ x_1 : B_1 <:^{+} \ y_1 : A_1 \rightsquigarrow J_1 \\ x_2 : A_2 <:^{+} \ y_2 : B_2 \rightsquigarrow J_2 \end{array}}{x : A_1 \to A_2 <:^{\pm} \ y : B_1 \to B_2 \rightsquigarrow \texttt{code}}$$

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

S-Rcd
$$\frac{\begin{array}{c} T_1 = \{\ell : |A|\} \\ T_2 = \{\ell : |B|\} \\ x_0 : A <:^{+} \ y_0 : B \rightsquigarrow J \end{array}}{x : \{\ell : A\} <:^{\pm} \ y : \{\ell : B\} \rightsquigarrow \texttt{code}}$$

S-AndL
$$\frac{x : A <:^{-} \ y : C \rightsquigarrow J}{x : A \,\&\, B <:^{\pm} \ y : C \rightsquigarrow J}$$

S-AndR
$$\frac{x : B <:^{-} \ y : C \rightsquigarrow J}{x : A \,\&\, B <:^{\pm} \ y : C \rightsquigarrow J}$$

```
/* 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 = {};  J1;
  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)*

M-And

$$\frac{}{z : A \ \triangleright \ z : A \,\&\, B \ \triangleleft \ z : B \rightsquigarrow \varnothing}$$

M-Arrow

$$\frac{\begin{array}{c} T = |\overrightarrow{B}| \\ T_1 = |\overrightarrow{B_1}| \qquad T_2 = |\overrightarrow{B_2}| \\ y_1 : B_1 \ \triangleright \ y : B \ \triangleleft \ y_2 : B_2 \rightsquigarrow J \end{array}}{x_1 : A \to B_1 \ \triangleright \ z : A \to B \ \triangleleft \ x_2 : A \to B_2 \rightsquigarrow \texttt{code}}$$

M-All

$$\frac{\begin{array}{c} T = |B|^{\forall} \\ T_1 = |B_1|^{\forall} \qquad T_2 = |B_2|^{\forall} \\ y_1 : B_1 \ \triangleright \ y : B \ \triangleleft \ y_2 : B_2 \rightsquigarrow J \end{array}}{x_1 : \forall X * A.\ B_1 \ \triangleright \ z : \forall X * A.\ B \ \triangleleft \ x_2 : \forall X * A.\ B_2 \rightsquigarrow \texttt{code}}$$

M-Rcd

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

```
/* 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;
});
```