Staging as a Mechanism for Algorithm Derivation

Nicolas Feltman

October 1, 2013

1 Introduction

Add intro in a bit.

2 $\lambda^{\mathbb{1}\mathbb{2}}$ Definition

In this section I define $\lambda^{\mathbb{1}\mathbb{2}}$, a two-stage language. Throughout this document, I will use $\mathbb{1}$ and $\mathbb{2}$ to refer to the two stages. A grammer for the terms, types, and contexts of $\lambda^{\mathbb{1}\mathbb{2}}$ is shown in Figure 1. Although both stages of $\lambda^{\mathbb{1}\mathbb{2}}$ contain products, sums, and functions, I have chosen to seperate the stages syntactically to emphasize that this need not be the case.

You'll notice that there are three seperate mechanisms by which one stage can refer to another: `next`, `prev`, and `save`. Specifically, `next` allows a stage $\mathbb{1}$ term to reference computation that will occur in the future, at stage $\mathbb{2}$, whereas `prev` and `save` allow a stage $\mathbb{2}$ computation to refer back to stage $\mathbb{1}$. The precise meaning of these constructs will be explored in future sections.

3 Typing

The rules relating types and terms are given in Figure 2. The structure of the core typing judgement is entirely standard. The nullary terms (currently just the unit value) are all explicitly annotated with a stage. The non-nullary terms essentially preserve the stage of their inputs, except for `save`, which transtions an expression from $\mathbb{1}$ to $\mathbb{2}$. From this, it is apparent that a stage $\mathbb{1}$ term cannot depend on any stage $\mathbb{2}$ terms. This property can be justified by the notion that the program gets "more reduced" as time goes on, and that a term can be no more reduced than those it depends on.

Stag has one additional peculiar feature. The predicate of a case statement can be stage $\mathbb{2}$ while the branches have subterms with stage $\mathbb{1}$. This implies that some computation in a branch must be performed even before we know which branch to take. This feature, called *speculation*, will be explored more later.

Figure 1: Stag Grammar

$$
\begin{aligned}
\langle \mathbb{1}\text{-}\mathit{type} \rangle &= \text{unit} \\
&\mid\ \langle \mathbb{1}\text{-}\mathit{type} \rangle \times \langle \mathbb{1}\text{-}\mathit{type} \rangle \\
&\mid\ \langle \mathbb{1}\text{-}\mathit{type} \rangle + \langle \mathbb{1}\text{-}\mathit{type} \rangle \\
&\mid\ \langle \mathbb{1}\text{-}\mathit{type} \rangle \to \langle \mathbb{1}\text{-}\mathit{type} \rangle \\
&\mid\ \bigcirc\, \langle \mathbb{2}\text{-}\mathit{type} \rangle \\
\langle \mathbb{2}\text{-}\mathit{type} \rangle &= \text{unit} \\
&\mid\ \langle \mathbb{2}\text{-}\mathit{type} \rangle \times \langle \mathbb{1}\text{-}\mathit{type} \rangle \\
&\mid\ \langle \mathbb{2}\text{-}\mathit{type} \rangle + \langle \mathbb{2}\text{-}\mathit{type} \rangle \\
&\mid\ \langle \mathbb{2}\text{-}\mathit{type} \rangle \to \langle \mathbb{2}\text{-}\mathit{type} \rangle \\
\langle \mathbb{1}\text{-}\mathit{exp} \rangle &= \lambda \langle \mathit{var} \rangle . \langle \mathbb{1}\text{-}\mathit{exp} \rangle \\
&\mid\ \langle \mathit{var} \rangle \\
&\mid\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle \\
&\mid\ () \\
&\mid\ (\langle \mathbb{1}\text{-}\mathit{exp} \rangle, \langle \mathbb{1}\text{-}\mathit{exp} \rangle) \\
&\mid\ \pi_1\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle \ \mid\ \pi_2\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle \\
&\mid\ \iota_1\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle \ \mid\ \iota_2\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle \\
&\mid\ \texttt{case}\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle\ \texttt{of}\ \langle \mathit{var} \rangle . \langle \mathbb{1}\text{-}\mathit{exp} \rangle\ \text{`}|\text{'}\ \langle \mathit{var} \rangle . \langle \mathbb{1}\text{-}\mathit{exp} \rangle \\
&\mid\ \texttt{next}\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle \\
\langle \mathbb{2}\text{-}\mathit{exp} \rangle &= \lambda \langle \mathit{var} \rangle . \langle \mathbb{2}\text{-}\mathit{exp} \rangle \\
&\mid\ \langle \mathit{var} \rangle \\
&\mid\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle \\
&\mid\ () \\
&\mid\ (\langle \mathbb{2}\text{-}\mathit{exp} \rangle, \langle \mathbb{2}\text{-}\mathit{exp} \rangle) \\
&\mid\ \pi_1\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle \ \mid\ \pi_2\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle \\
&\mid\ \iota_1\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle \ \mid\ \iota_2\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle \\
&\mid\ \texttt{case}\ \langle \mathbb{2}\text{-}\mathit{exp} \rangle\ \texttt{of}\ \langle \mathit{var} \rangle . \langle \mathbb{2}\text{-}\mathit{exp} \rangle\ \text{`}|\text{'}\ \langle \mathit{var} \rangle . \langle \mathbb{2}\text{-}\mathit{exp} \rangle \\
&\mid\ \texttt{save}\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle \\
&\mid\ \texttt{prev}\ \langle \mathbb{1}\text{-}\mathit{exp} \rangle \\
\langle \mathit{cont} \rangle &= \cdot \\
&\mid\ \langle \mathit{cont} \rangle, \langle \mathit{var} \rangle : \langle \mathbb{1}\text{-}\mathit{type} \rangle^{\mathbb{1}} \\
&\mid\ \langle \mathit{cont} \rangle, \langle \mathit{var} \rangle : \langle \mathbb{2}\text{-}\mathit{type} \rangle^{\mathbb{2}}
\end{aligned}
$$

Figure 2: Typing Rules

$$\frac{\cdot}{\Gamma \vdash^{\sigma} () : \text{unit}} \tag{1}$$

$$\frac{\Gamma \vdash^{\sigma} e_1 : A \quad \Gamma \vdash^{\sigma} e_2 : B}{\Gamma \vdash^{\sigma} (e_1, e_2) : A \times B} \tag{2}$$

$$\frac{\Gamma \vdash^{\sigma} e : A \times B}{\Gamma \vdash^{\sigma} \pi_1\ e : A} \tag{3}$$

$$\frac{\Gamma \vdash^{\sigma} e : A \times B}{\Gamma \vdash^{\sigma} \pi_2\ e : B} \tag{4}$$

$$\frac{\Gamma, x : A^{\sigma} \vdash e : B}{\Gamma \vdash^{\sigma} (\lambda x : A.e) : A \to B} \tag{5}$$

$$\frac{x : A^{\sigma} \in \Gamma}{\Gamma \vdash^{\sigma} x : A} \tag{6}$$

$$\frac{\Gamma \vdash^{\sigma} e_1 : A \to B \quad \Gamma \vdash^{\sigma} e_2 : A}{\Gamma \vdash^{\sigma} e_1\ e_2 : B} \tag{7}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e : A}{\Gamma \vdash^{\mathbb{2}} \texttt{save}\ e : A} \tag{8}$$

$$\frac{\Gamma \vdash^{\mathbb{2}} e : A}{\Gamma \vdash^{\mathbb{1}} \texttt{next}\ e : \bigcirc A} \tag{9}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e : \bigcirc A}{\Gamma \vdash^{\mathbb{2}} \texttt{prev}\ e : A} \tag{10}$$

3.1 Evaluation

The evaluation is given in Figure 3. Were it complete, it would contain a definition of values (which we predict to be unchanged by the staging system), and a big-step sematics relating terms to values. The big-step evaluation should be indexed by the stage at which it completes. That is, we have both $\Downarrow_{\mathbb{1}}$ and $\Downarrow_{\mathbb{2}}$. Also, the bigstep semantics will reflect that speculation occurs down both branches of a case as part of $\Downarrow_{\mathbb{1}}$ reduction.

3.2 Stage Splitting

The core operation of interest is the process of "stage splitting," wherein a term with stage $\mathbb{2}$ is converted to a precomputed part, and a residual that depends on that precomputed part. Specifically, we introduce a judgement "$\Gamma \vdash e \overset{\mathbb{2}}{\rightsquigarrow} [p, x.r]$", which can be read "under the context Γ, e stage-splits into a precomputation p, and a residual r which is open on x". The idea is that p contains the parts of e that are stage $\mathbb{1}$, r contains the parts of e that are stage $\mathbb{2}$, and the reduced value of p is bound to x when evaluating r.

Note that the precomputation and residual are not actually terms in Stag. They are terms in a simpler language, called Doe, which is essentially Stag without the staging constructs. The grammar for Doe is shown in Figure **??**. The typing rules for Doe are not shown, but can be guessed.

The full rules for splitting are shown in Figure **??**. In two of the rules, I reference another judgement, of the form "$\Gamma \vdash e \overset{\mathbb{1}}{\rightsquigarrow} e'$". This means that the one-stage Stag term e simply translates to a Doe term e'. There are no surprises in how this works, so the details are omitted.

Even without a semantics done, we can think of two theorems that should hold true of stage splitting. Namely, that good types in lead to good types out. Explicitly:

$$\begin{array}{ll}
\text{If } \Gamma \vdash e : \tau \ @\ \mathbb{1} & \qquad\text{If } \Gamma \vdash e : \tau \ @\ \mathbb{2} \\
\text{then } \Gamma \vdash e \overset{\mathbb{1}}{\rightsquigarrow} e' & \qquad\text{then } \Gamma \vdash e \overset{\mathbb{2}}{\rightsquigarrow} [p, x.r] \\
\text{and } \Gamma \vdash e' : \tau & \qquad\text{and } \Gamma_{\mathbb{1}} \vdash p : \tau' \\
 & \qquad\text{and } \Gamma_{\mathbb{2}}, x : \tau' \vdash r : \tau
\end{array}$$

Note that I'm implicitly abusing the identical structures of pretypes in Stag and types in Doe. Also, for any stage σ, we define Γ_σ as,

$$(\cdot)_\sigma = \cdot \tag{25}$$

$$(\Gamma, x : \tau \ @\ \sigma)_\sigma = \Gamma_\sigma, \tau \tag{26}$$

$$(\Gamma, x : \tau \ @\ \sigma')_\sigma = \Gamma_\sigma \qquad \text{where } \sigma \neq \sigma' \tag{27}$$

3.3 Implementation

I have an implementation of stage-splitting in SML. Currently, the code only keeps track of stage, and not the pre-type.

[Add results.]

Figure 3: Typing Rules

$$\frac{\cdot}{()_\sigma \Downarrow_\sigma ()} \tag{11}$$

$$\frac{e \Downarrow_\sigma v \quad f\ v \Downarrow v'}{f\ e \Downarrow_\sigma v'} \tag{12}$$

$$\frac{e \Downarrow_\sigma (v_1, v_2)}{\pi_1\ e \Downarrow_\sigma v_1} \tag{13}$$

$$\frac{e \Downarrow_\sigma (v_1, v_2)}{\pi_2\ e \Downarrow_\sigma v_2} \tag{14}$$

$$\frac{e \Downarrow_\sigma v}{\iota_1\ e \Downarrow_\sigma \iota_1\ v} \tag{15}$$

$$\frac{e \Downarrow_\sigma v}{\iota_2\ e \Downarrow_\sigma \iota_2\ v} \tag{16}$$

$$\frac{e_1 \Downarrow_\sigma v_1 \quad e_2 \Downarrow_\sigma v_2}{(e_1, e_2) \Downarrow_\sigma (v_1, v_2)} \tag{17}$$

$$\frac{e_1 \Downarrow_\sigma v_1 \quad [v'/x]e_2 \Downarrow_\sigma v_2}{\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \Downarrow_\sigma v_2} \tag{18}$$

$$\frac{e_1 \Downarrow_{\mathbb{1}} v_1 \quad [v'/x]e_2 \Downarrow_p v_2}{\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \Downarrow_p v_2} \tag{19}$$

$$\frac{e_1 \Downarrow_p v_1 \quad e_2 \Downarrow_p v_2}{\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \Downarrow_p \texttt{let}\ x = v_1\ \texttt{in}\ v_2} \tag{20}$$

$$\frac{e \Downarrow_\sigma (\iota_i\ v) \quad [v/x_i]e_i \Downarrow_\sigma v}{\texttt{case}\ e\ \texttt{of}\ x_1.e_1 \mid x_2.e_2 \Downarrow_\sigma v} \tag{21}$$

$$\frac{e \Downarrow_{\mathbb{1}} (\iota_i\ v) \quad [v/x_i]e_i \Downarrow_p v}{\texttt{case}\ e\ \texttt{of}\ x_1.e_1 \mid x_2.e_2 \Downarrow_p v} \tag{22}$$

$$\frac{e \Downarrow_p v \quad e_1 \Downarrow_p v_1 \quad e_2 \Downarrow_p v_2}{\texttt{case}\ e\ \texttt{of}\ x_1.e_1 \mid x_2.e_2 \Downarrow_p \texttt{case}\ v\ \texttt{of}\ x_1.v_1 \mid x_2.v_2} \tag{23}$$

$$\frac{e : A\ @\ \mathbb{1}}{\texttt{save}\ e \Downarrow_{\mathbb{2}}} \tag{24}$$

Figure 4: Basic Splitting

$$\frac{\Gamma \vdash^{\mathbb{2}} e : A \overset{\mathbb{2}}{\leadsto} [p, l.r]}{\Gamma \vdash^{\mathbb{1}} \texttt{next}\ e : \bigcirc A \overset{\mathbb{1}}{\leadsto} [((), p), l.r]} \tag{28}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e : \bigcirc A \overset{\mathbb{1}}{\leadsto} [c, l.r]}{\Gamma \vdash^{\mathbb{2}} \texttt{prev}\ e : A \overset{\mathbb{2}}{\leadsto} [\pi_2\ c, l.r]} \tag{29}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e : A \quad A \text{ transferable}}{\Gamma \vdash^{\mathbb{2}} \texttt{save}\ e : A \overset{\mathbb{2}}{\leadsto} [e, l.l]} \tag{30}$$

Figure 5: Product Splitting

$$\frac{\cdot}{\Gamma \vdash^{\mathbb{1}} () : \text{unit} \overset{\mathbb{1}}{\leadsto} [((), ()), \_.()]} \tag{31}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e_1 : A \overset{\mathbb{1}}{\leadsto} [c_1, l_1.r_1] \quad \Gamma \vdash^{\mathbb{1}} e_2 : B \overset{\mathbb{1}}{\leadsto} [c_2, l_2.r_2]}{\Gamma \vdash^{\mathbb{1}} (e_1, e_2) : A \times B \overset{\mathbb{1}}{\leadsto} [((\pi_1 c, \pi_1 c), (\pi_2 c_1, \pi_2 c_2)), l.(\texttt{let}\ l_1 = \pi_1 l\ \texttt{in}\ r_1, \texttt{let}\ l_2 = \pi_2 l\ \texttt{in}\ r_2)]} \tag{32}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e : A \times B \overset{\mathbb{1}}{\leadsto} [c, l.r]}{\Gamma \vdash^{\mathbb{1}} \pi_1\ e : A \overset{\mathbb{1}}{\leadsto} [(\pi_1\ p), l.\pi_1\ r]} \tag{33}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e : A \times B \overset{\mathbb{1}}{\leadsto} [c, l.r]}{\Gamma \vdash^{\mathbb{1}} \pi_2\ e : B \overset{\mathbb{1}}{\leadsto} [\pi_2\ p, l.\pi_2\ r]} \tag{34}$$

$$\frac{\cdot}{\Gamma \vdash^{\mathbb{2}} () : \text{unit} \overset{\mathbb{2}}{\leadsto} [(), \_.()]} \tag{35}$$

$$\frac{\Gamma \vdash^{\mathbb{2}} e_1 : A \overset{\mathbb{2}}{\leadsto} [p_1, l_1.r_1] \quad \Gamma \vdash^{\mathbb{2}} e_2 : B \overset{\mathbb{2}}{\leadsto} [p_2, l_2.r_2]}{\Gamma \vdash^{\mathbb{2}} (e_1, e_2) : A \times B \overset{\mathbb{2}}{\leadsto} [(p_1, p_2), l.(\texttt{let}\ l_1 = \pi_1\ l\ \texttt{in}\ r_1, \texttt{let}\ l_2 = \pi_2\ l\ \texttt{in}\ r_2)]} \tag{36}$$

$$\frac{\Gamma \vdash^{\mathbb{2}} e : A \times B \overset{\mathbb{2}}{\leadsto} [p, l.r]}{\Gamma \vdash^{\mathbb{2}} \pi_1\ e : A \overset{\mathbb{2}}{\leadsto} [p, l.\pi_1\ r]} \tag{37}$$

$$\frac{\Gamma \vdash^{\mathbb{2}} e : A \times B \overset{\mathbb{2}}{\leadsto} [p, l.r]}{\Gamma \vdash^{\mathbb{2}} \pi_2\ e : B \overset{\mathbb{2}}{\leadsto} [p, l.\pi_2\ r]} \tag{38}$$

Figure 6: Function Splitting

$$\frac{x : A^{\mathbb{1}} \in \Gamma}{\Gamma \vdash^{\mathbb{1}} x : A \overset{\mathbb{1}}{\rightsquigarrow} [(x, ()), \_.x]} \tag{39}$$

$$\frac{\Gamma, x : A^{\mathbb{1}} \vdash e : B \overset{\mathbb{1}}{\rightsquigarrow} [c, l.r]}{\Gamma \vdash^{\mathbb{1}} (\lambda x : A.e) : A \to B \overset{\mathbb{1}}{\rightsquigarrow} [(\lambda x : |A|_{\mathbb{1}}.c, ()), \_.(\lambda(x, l) : |A|_{\mathbb{2}} \times \tau.r)]} \tag{40}$$

$$\frac{\Gamma \vdash^{\mathbb{1}} e_1 : A \to B \overset{\mathbb{1}}{\rightsquigarrow} [c_1, l_1.r_1] \quad \Gamma \vdash^{\mathbb{1}} e_2 : A \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2]}{\Gamma \vdash^{\mathbb{1}} e_1\ e_2 : B \overset{\mathbb{1}}{\rightsquigarrow} \left[\begin{array}{l} \texttt{let}\ y = (\pi_1 c_1)(\pi_1 c2)\ \texttt{in}\ (\pi_1 y, (\pi_2 c_1, \pi_2 c_2, \pi_2 y)), \\ l.(\texttt{let}\ x_1 = \pi_1\ l\ \texttt{in}\ r_1)(\texttt{let}\ x_2 = \pi_2\ l\ \texttt{in}\ r_2, \pi_3\ l) \end{array}\right]} \tag{41}$$

$$\frac{x : A^{\mathbb{2}} \in \Gamma}{\Gamma \vdash^{\mathbb{2}} x : A \overset{\mathbb{2}}{\rightsquigarrow} [(), \_.x]} \tag{42}$$

$$\frac{\Gamma, x : A^{\mathbb{2}} \vdash e : B \overset{\mathbb{2}}{\rightsquigarrow} [p, l.r]}{\Gamma \vdash^{\mathbb{2}} (\lambda x : A.e) : A \to B \overset{\mathbb{2}}{\rightsquigarrow} [p, l.\lambda x : A.r]} \tag{43}$$

$$\frac{\Gamma \vdash^{\mathbb{2}} e_1 : A \to B \overset{\mathbb{2}}{\rightsquigarrow} [p_1, l_1.r_1] \quad \Gamma \vdash^{\mathbb{2}} e_2 : A \overset{\mathbb{2}}{\rightsquigarrow} [p_2, l_2.r_2]}{\Gamma \vdash^{\mathbb{2}} e_1\ e_2 : B \overset{\mathbb{2}}{\rightsquigarrow} [(p_1, p_2), l.(\texttt{let}\ x_1 = \pi_1\ l\ \texttt{in}\ r_1)(\texttt{let}\ x_2 = \pi_2\ l\ \texttt{in}\ r_2)]} \tag{44}$$

Figure 7: Sum Splitting

$$\frac{\Gamma \vdash^{\mathbb{1}} e_1 : A + B \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2] \quad \Gamma, x_2 : A^{\mathbb{1}} \vdash^{\mathbb{1}} e_2 : C \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2] \quad \Gamma, x_3 : B^{\mathbb{1}} \vdash^{\mathbb{1}} e_3 : C \overset{\mathbb{1}}{\rightsquigarrow} [c_3, l_3.r_3]}{\Gamma \vdash^{\mathbb{1}} \texttt{case}\ e_1\ \texttt{of}\ x_2.e_2 \mid x_3.e_3 : C \overset{\mathbb{1}}{\rightsquigarrow} \left[\left(\begin{array}{l} \texttt{case}\ \pi_1 c_2\ \texttt{of} \\ \quad x_2.\texttt{let}\ y = c_2\ \texttt{in}\ (\pi_1 y, \iota_1(\pi_2 y)) \\ \mid x_3.\texttt{let}\ y = c_3\ \texttt{in}\ (\pi_1 y, \iota_2(\pi_2 y)) \end{array}\right), l. \left(\begin{array}{c} \texttt{case}\ l\ \texttt{of} \\ l_2.r_2 \\ \mid l_3.r_3 \end{array}\right)\right]} \tag{45}$$

$$\frac{\Gamma \vdash^{\mathbb{2}} e_1 : A + B \overset{\mathbb{2}}{\rightsquigarrow} [p_1, l_1.r_1] \quad \Gamma, x_2 : A^{\mathbb{2}} \vdash^{\mathbb{2}} e_2 : C \overset{\mathbb{2}}{\rightsquigarrow} [p_2, l_2.r_2] \quad \Gamma, x_3 : B^{\mathbb{2}} \vdash^{\mathbb{2}} e_3 : C \overset{\mathbb{2}}{\rightsquigarrow} [p_3, l_3.r_3]}{\Gamma \vdash^{\mathbb{2}} \texttt{case}\ e_1\ \texttt{of}\ x_2.e_2 \mid x_3.e_3 : C \overset{\mathbb{2}}{\rightsquigarrow} \left[(p_1, (p_2, p_3)), l. \left(\begin{array}{c} \texttt{case}\ (\texttt{let}\ l_1 = \pi_1 l\ \texttt{in}\ r_1)\ \texttt{of} \\ x_2.\texttt{let}\ l_2 = \pi_1(\pi_2 l)\ \texttt{in}\ r_2 \\ \mid x_3.\texttt{let}\ l_3 = \pi_2(\pi_2 l)\ \texttt{in}\ r_3 \end{array}\right)\right]} \tag{46}$$

4 Related Work

4.1 Partial Evaluation

One can immediately see a connection between this work and partial evaluation. Both involve the idea of specializing a piece of code to some of it's inputs, leaving a residual that depends only on the remaining inputs. But stage splitting is actually only part of partial evaluation. At it's core, stage splitting is "factor out the first stage, reduce it to a value, and express the second stage abstractly over that value," whereas partial evaluation is "factor out the first stage, reduce it to a value, and specialize the code of the second stage to that value." Expressed equationally,

$$\begin{array}{rll} \textbf{Partial Evaluation}: & p(f,a) = f_a & \text{s.t. } f_a(b) = f(a,b) \\ \textbf{Stage Splitting}: & s(f) = (f_1, f_2) & \text{s.t. } f_2(f_1(a), b) = f(a,b) \end{array}$$

Immediately, we note that stage splitting has broader application than partial evaluation, since the latter requires that p (specifically, some code-specializing apparatus) and a be available at the same time. If this requirement is satisfied, then we can compare apples to apples by creating a partial evaluator out of a stage splitter:

$$f_a = (\texttt{let } x = f_1(a) \texttt{ in } \lambda b.f_2(x,b))$$

From this view, we see that partial evaluators are more powerful because they can avoid memeory loads, prune unused branches, and even duplicate recursive code for further specialization. These are largely free wins, except for the recursive code generation, which might have large space costs.

4.2 Temporal Logic

There are two papers (called "circle" and "square") by Rowan Davies which give a justification for binding time analysis in modal logics. Those papers feel stylistically similar to this work. Circle, in addition to its other contributions, gives a particularly good account of the difference between its style (which we use), and the rest of the literature.

There's an important question that our work will need to answer about the difference between the logic that our type system induces and those of Davies. In particular, what does it mean that our language has no `prev` operator?

4.3 Speculation

I can find nothing in the literature that looks like our speculation. That's probably because speculation is unsafe (especially around side effects), and so it would be a terrible idea in a system without lots of programmer direction.