COMP 302 – Programming Languages and Paradigms
Classtest 3 Practice Problems

Victor Desautels & Jacob Thomas Errington

Winter 2025

Static Semantics

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \text{yes } e : \text{Maybe } A} \qquad \frac{}{\Gamma \vdash \text{no } A : \text{Maybe } A}$$

The A attached to `no` functions as a type annotation, specifying what kind of failure, i.e., a "failure to produce a value of type A."

$$\frac{\Gamma \vdash e_1 : \text{Maybe } A \quad \Gamma \vdash e_2 : A}{\Gamma \vdash \text{try } e_1 \text{ with } e_2 : A}$$

The `try` syntax is used to handle a potentially failing computation e_1 by providing an alternative computation e_2.

$$\frac{\Gamma \vdash e_1 : \text{Maybe } A \quad \Gamma, x : A \vdash e_2 : \text{Maybe } B}{\Gamma \vdash \text{let yes } x = e_1 \text{ in } e_2 : \text{Maybe } B}$$

The `let-yes` syntax is used to chain potentially-failing computations together.

Operational Semantics

$$\frac{}{\text{no } A \Downarrow \text{no } A} \qquad \frac{e \Downarrow v}{\text{yes } e \Downarrow \text{yes } v} \qquad \frac{e_1 \Downarrow \text{no } A}{\text{let yes } x = e_1 \text{ in } e_2 \Downarrow \text{no } A}$$

That rightmost rule is actually incorrect. Recall that evaluation preserves types. The expression we start with `let yes x = e1 in e2` has type `Maybe B`, so when `e1` evaluates to `no A`, we really should evaluate to `no B` in the conclusion. However, we don't have a type annotation `B` on hand to use. So we 'cheat' and use the wrong type annotation. The annotation never actually plays a role during evaluation, so this doesn't cause any real harm.

$$\frac{e_1 \Downarrow \text{yes } v \quad [v/x]e_2 \Downarrow v'}{\text{let yes } x = e_1 \text{ in } e_2 \Downarrow v'} \qquad \frac{e_1 \Downarrow \text{yes } v}{\text{try } e_1 \text{ with } e_2 \Downarrow v} \qquad \frac{e_1 \Downarrow \text{no } A \quad e_2 \Downarrow v}{\text{try } e_1 \text{ with } e_2 \Downarrow v}$$