Statics

The Phases of PLs

Static Phase: Well-formedness of programs; e.g., parsing and type checking.

Dynamic Phase: Execution of programs.

Statics

A collection of rules, called typing rules, for deriving <u>typing judgments</u> stating that an expression is a well-formed for a certain type.

Statics

Types mediate interaction between the various constituent parts of a program by predicting the execution behavior of the parts so that we are sure that they fit together properly at run time.

Statics: Example

Suppose we have a function `f`, and an input `x`, then what can we say about the application `f(x)`?

Statics: Example

Suppose we have a function `f : Int -> Int`, and an input `x:Int`, then what can we say about the application "`f(x)`"?

Statics: Syntax

Typ	τ	$::=$	$\mathtt{num}$	$\mathtt{num}$	numbers
			$\mathtt{str}$	$\mathtt{str}$	strings
Exp	e	$::=$	x	x	variable
			$\mathtt{num}[n]$	n	numeral
			$\mathtt{str}[s]$	$''s''$	literal
			$\mathtt{plus}(e_1;e_2)$	$e_1 + e_2$	addition
			$\mathtt{times}(e_1;e_2)$	$e_1 * e_2$	multiplication
			$\mathtt{cat}(e_1;e_2)$	e_1 ^ e_2	concatenation
			$\mathtt{len}(e)$	$\lvert e \rvert$	length
			$\mathtt{let}(e_1;x.e_2)$	$\mathtt{let}\, x\, \mathtt{be}\, e_1\, \mathtt{in}\, e_2$	definition

Statics: Typing

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ \text{Var}$$

$$\frac{}{\Gamma \vdash \mathtt{str}[s] : \mathtt{str}}\ \text{Str}$$

$$\frac{}{\Gamma \vdash \mathtt{num}[n] : \mathtt{num}}\ \text{Num}$$

$$\frac{\Gamma \vdash e_1 : \mathtt{num} \quad \Gamma \vdash e_2 : \mathtt{num}}{\Gamma \vdash \mathtt{plus}(e_1;e_2) : \mathtt{num}}\ \text{Plus}$$

$$\frac{\Gamma \vdash e_1 : \mathtt{num} \quad \Gamma \vdash e_2 : \mathtt{num}}{\Gamma \vdash \mathtt{times}(e_1;e_2) : \mathtt{num}}\ \text{Times}$$

$$\frac{\Gamma \vdash e_1 : \mathtt{str} \quad \Gamma \vdash e_2 : \mathtt{str}}{\Gamma \vdash \mathtt{cat}(e_1;e_2) : \mathtt{str}}\ \text{Cat}$$

$$\frac{\Gamma \vdash e : \mathtt{str}}{\Gamma \vdash \mathtt{len}(e) : \mathtt{num}}\ \text{Len}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \mathtt{let}(e_1;x.e_2) : \tau_2}\ \text{Let}$$

Statics: Properties

Lemma (Unicity of Typing): For every context Γ and expression e, there exists at most one τ such that $\Gamma \vdash e : \tau$.

Statics: Properties

Lemma (Inversion for Typing): Suppose $\Gamma \vdash e : \tau$.

1. If $e = \mathsf{plus}(e_1; e_2)$, then $\tau = \mathsf{num}$, $\Gamma \vdash e_1 : \mathsf{num}$, and $\Gamma \vdash e_2 : \mathsf{num}$.

2. If $e = \mathsf{times}(e_1; e_2)$, then $\tau = \mathsf{num}$, $\Gamma \vdash e_1 : \mathsf{num}$, and $\Gamma \vdash e_2 : \mathsf{num}$.

3. If $e = \mathsf{cat}(e_1; e_2)$, then $\tau = \mathsf{str}$, $\Gamma \vdash e_1 : \mathsf{str}$, and $\Gamma \vdash e_2 : \mathsf{str}$.

4. If $e = \mathsf{len}(e_1)$, then $\tau = \mathsf{num}$, $\Gamma \vdash e_1 : \mathsf{str}$.

5. If $e = \mathsf{let}(e_1; x . e_2)$, then $\Gamma \vdash e_1 : \tau_1$ and $\Gamma, x : \tau_1 \vdash e_2 : \tau$