~minsheet~

# Static vs. Dynamics

We are going to split the life of a compute program into two phases

- static = anything that happens before running the program

  (compile time)

  e.g. parsing, static analysis, type checking, lexing, macros, optimization, compilation

  staging

- dynamic = everything that happens when a program is running

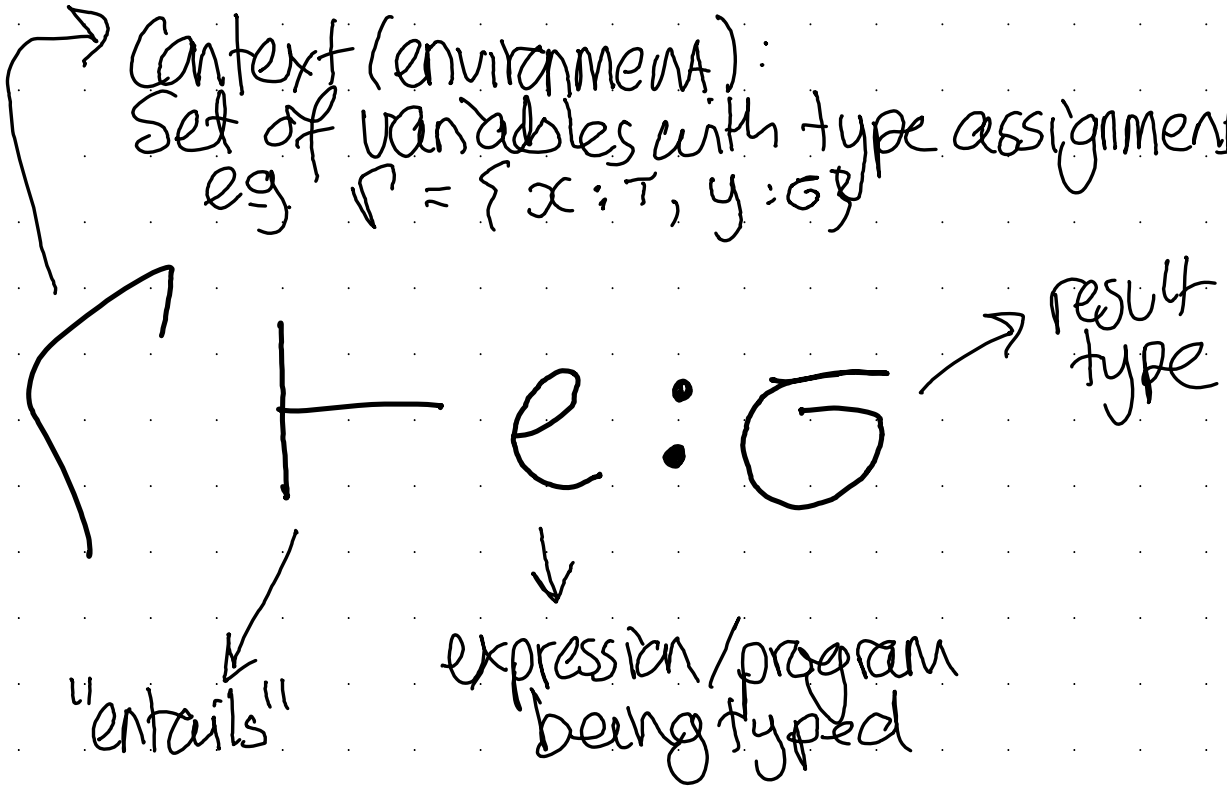
  (runtime)

  e.g. calculating the result, side-effects (IO, exceptions)

  hurts planet.

  Mems

# Typing Judgements




terms = well-typed

preterm = unproven to be well typed

Terms = well-typed preterms

# Binders

Binder = a language construct that closes over or binds variables

e.g. $\forall$ or $\exists$ $\forall x.$

let let $x = \ldots$ in

$\nearrow$ $\Rightarrow$

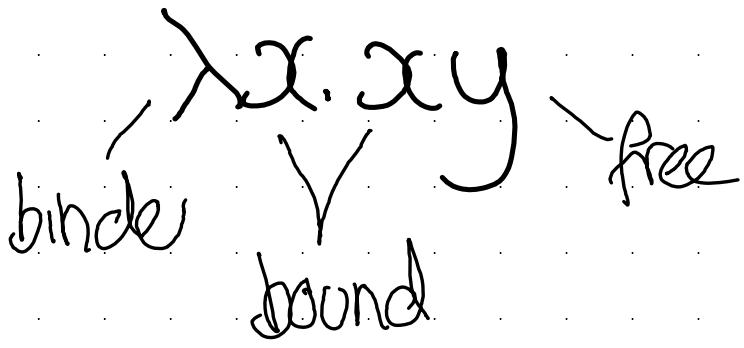
sometimes pattern matching

$\lim_{n \to \infty}$

$\sum$ $\prod$ ...

Bound variables = variables bound by binder

Free variables = variables not bound by a binder




Closed expr = one with no free vars
Open expr = one with free vars

open : $\lambda x.xy$

$\lambda x.z$

closed : $\lambda t.t$

$\lambda y.yy$

$\alpha$-equivalence / convertibility states that the name of a bound variable doesn't matter

i.e. $\lambda x.x =_\alpha \lambda y.y$

$\lambda a.ab =_\alpha \lambda x.xb$

$\lambda xy.y =_\alpha \lambda yx.x$

# A little language of numbers + strings - Syntax

## Backus-Naur form (BNF)

$$\tau ::= \text{Num} \mid \text{Str}$$

$\tau$ — syntactic category; Num, Str — concrete syntax; | — bars allowed!

**Abstract syntax** = the syntax of the form as it should appear in the abstract syntax tree (AST)

**Concrete syntax** = user-friendly abbreviation of the abstract syntax

$$e ::= x$$
$$\text{num}[n]$$
$$\text{str}[s]$$
$$\text{plus}(e_1; e_2) \quad e_1 + e_2$$
$$\text{times}(e_1; e_2) \quad e_1 * e_2$$
$$\text{cat}(e_1; e_2) \quad e_1 \mathbin{+\!\!+} e_2$$
$$\text{len}(e) \quad |e|$$
$$\text{let}(e_1; x.e_2) \quad \text{let } x \leftarrow e_1 \text{ in } e_2$$

"recursive" calls

↑ abstract syntax

↑ concrete syntax here

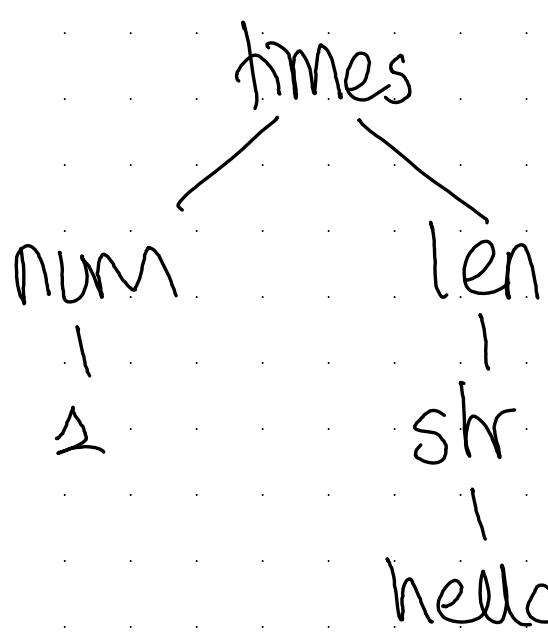
$n \in \mathbb{N}$

$s \in \Sigma^*$

e.g. $\Sigma = \{a, b, c \ldots\}$

e.g. $"11" \in \Sigma^*$ $"hello" \in \Sigma^*$

$$\text{times}(\text{num}[1]; \text{len}(\text{str}[\text{hello}]))$$



# A little language of numbers + strings – Typing!

$$\text{VAR}\ \frac{}{\Gamma, x:\sigma \vdash x:\sigma}$$

linear:

$$\text{VAR}\ \frac{}{x:\sigma \vdash x:\sigma}$$

$$\text{NUM}\ \frac{n \in \mathbb{N}}{\Gamma \vdash \text{num}[n] : \text{Num}}$$

$$\text{STR}\ \frac{s \in \Sigma^*}{\Gamma \vdash \text{str}[s] : \text{Str}}$$

$$\text{PLUS}\ \frac{\Gamma \vdash e_1 : \text{Num} \quad \Gamma \vdash e_2 : \text{Num}}{\Gamma \vdash \text{plus}(e_1; e_2) : \text{Num}}$$

$$\text{TIMES}\ \frac{\Gamma \vdash e_1 : \text{Num} \quad \Gamma \vdash e_2 : \text{Num}}{\Gamma \vdash \text{times}(e_1; e_2) : \text{Num}}$$

$$\text{CAT}\ \frac{\Gamma \vdash e_1 : \text{Str} \quad \Gamma \vdash e_2 : \text{Str}}{\Gamma \vdash \text{cat}(e_1; e_2) : \text{Str}}$$

$$\text{LEN}\ \frac{\Gamma \vdash e : \text{Str}}{\Gamma \vdash \text{len}(e) : \text{Num}}$$