Formal Semantics ← $Z, S(Z)$, $S(S(Z))$, ...

What is a PL?

unamb. (precise).

- BNF (Backus Naur Form)

$$n ::= \overset{\text{zero}}{Z} \mid \overset{\text{succ}}{S}(n)$$

Nat numbers, eg of formal syntax

"Judgements" — Ind.-def. relations via inference rules

$z \qquad S(z) \qquad S(S(z))$

$$z \qquad \begin{array}{c} S \\ | \\ z \end{array} \qquad \begin{array}{c} S \\ | \\ S \\ | \\ z \end{array} \qquad \begin{array}{cc} S & 1 \\ | & + \\ S & 1 \\ | & + \\ S & 1 \\ | & + \\ z & 0 \end{array}$$

$$\frac{n ::= \underline{z} \mid \underline{s(n)}}{}$$

"type"

n_1 n_2 S

n n'

n_1' n_2' $\Downarrow$

$$\frac{}{\boxed{\text{plus}(n_1, n_2) \Downarrow n_3}}$$

syntax for a relation

aka a ~~a~~ <u>judgement</u>

inf. rules are the constructors of the jud.

$$n ::= \underline{z} \mid \underline{s(n)}$$

$$\frac{}{plus(z, n_2) \Downarrow n_2}\ \text{Zero}$$

$$\frac{plus(n_1', n_2) \Downarrow n_3'}{plus(n_1, n_2) \Downarrow S(n_3')}\ \text{Succ} \qquad n_1 = S(n_1')$$

$$n_1 = \underbrace{\underline{S}(n_1')}_{n_1}$$

$$\frac{plus(n_1', n_2) \Downarrow n_3}{plus(\underbrace{s(n_1')}_{n_1}, n_2) \Downarrow s(n_3')}$$

$1 + 2 = 3$

$$\mathrm{plus}(\underbrace{s(z)}_{1}, \underbrace{s(s(z))}_{2}) \Downarrow \underbrace{s(s(s(z)))}_{3}$$

TODO: **Derive** this

Derivation (parse tree of the jud.)

$n_1 = S(n_1')$ $\quad n_3 = S(n_3')$

$$\frac{\boxed{\mathrm{plus}(n_1', n_2) \Downarrow n_3'}}{\mathrm{plus}(n_1, n_2) \Downarrow n_3}\ \mathrm{Succ}$$

$$\frac{\text{derivation}}{\text{(tree)}}$$

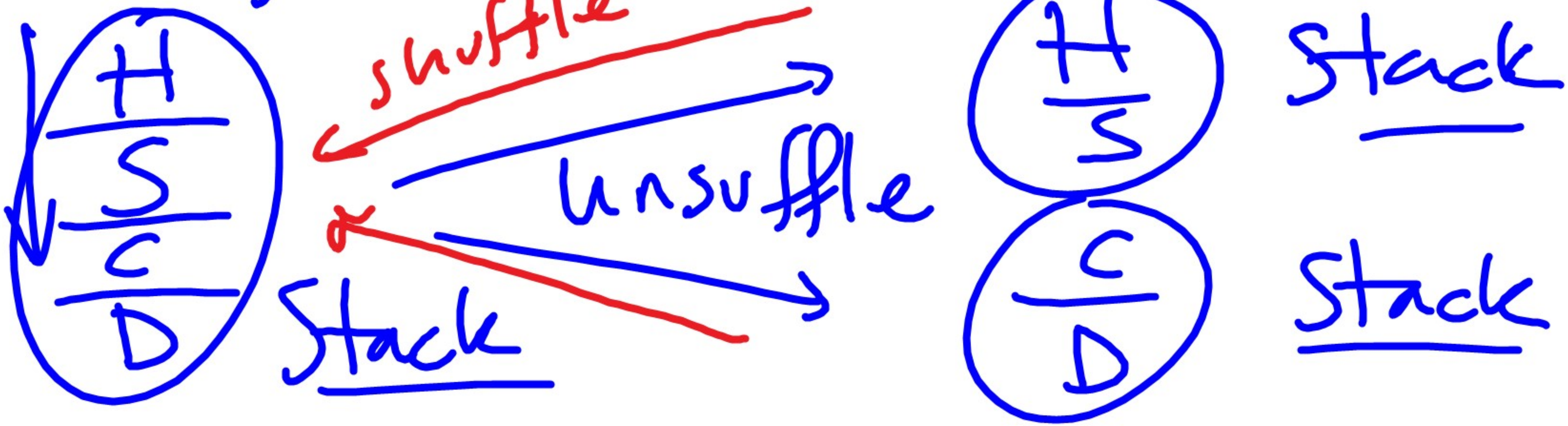
proof that $1 + 2 = 3$

$$\dfrac{\dfrac{}{\mathrm{plus}(\underline{0}, 2) \Downarrow \underline{2}}\ \mathrm{zero}}{\mathrm{plus}(1, 2) \Downarrow 3}\ \mathrm{Succ}$$

$C ::= \heartsuit \mid \spadesuit \mid \clubsuit \mid \diamondsuit$

$$\boxed{C ::= H \mid S \mid C \mid D}$$

BNF def for "card suit"



relations / functions

$$\text{unshuff}(s_1, \underbrace{s_2, s_3}_{\text{outputs.}})$$

inp

$C ::= H \mid S \mid D \mid C$

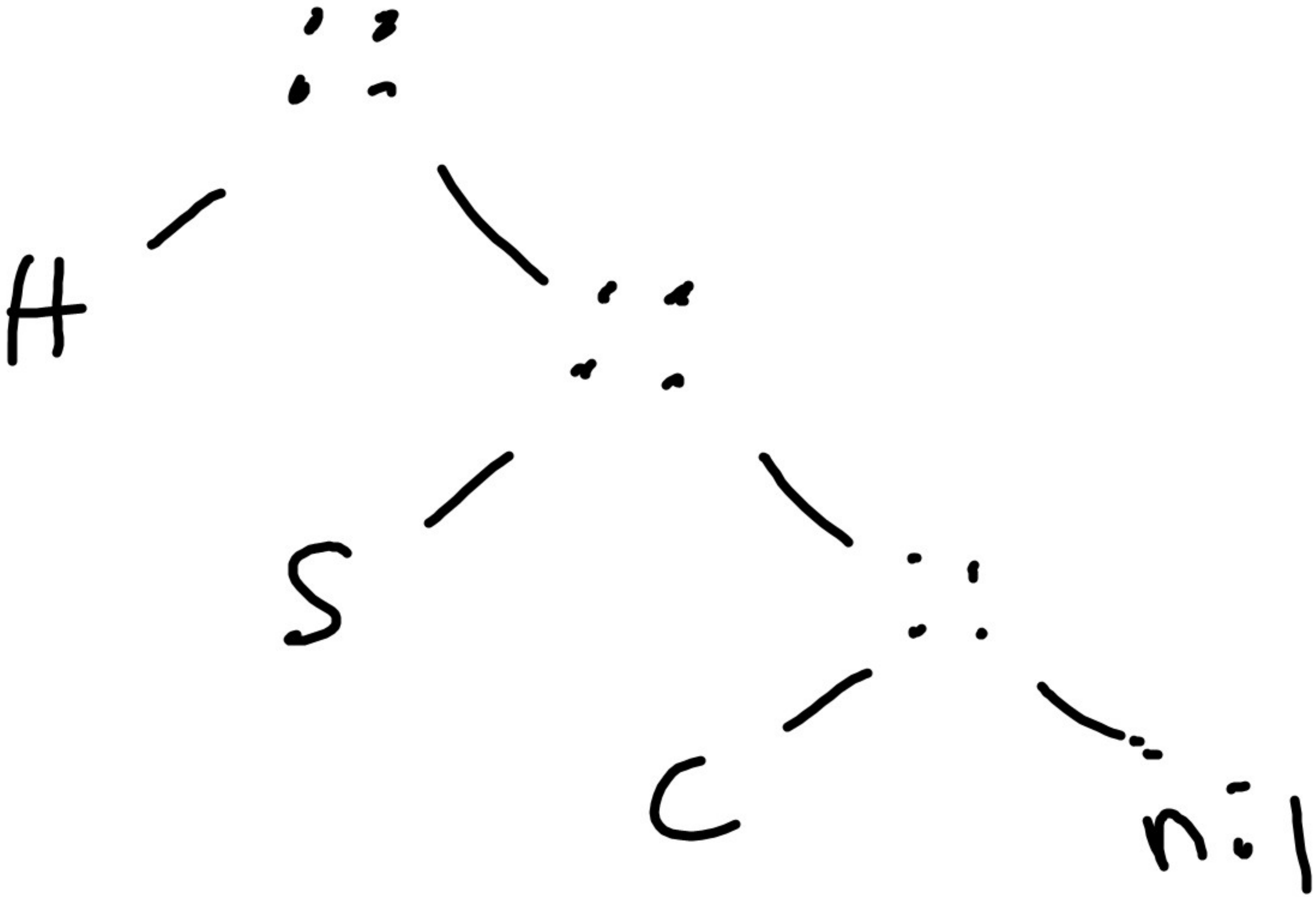
$S ::= nil \mid C :: S$

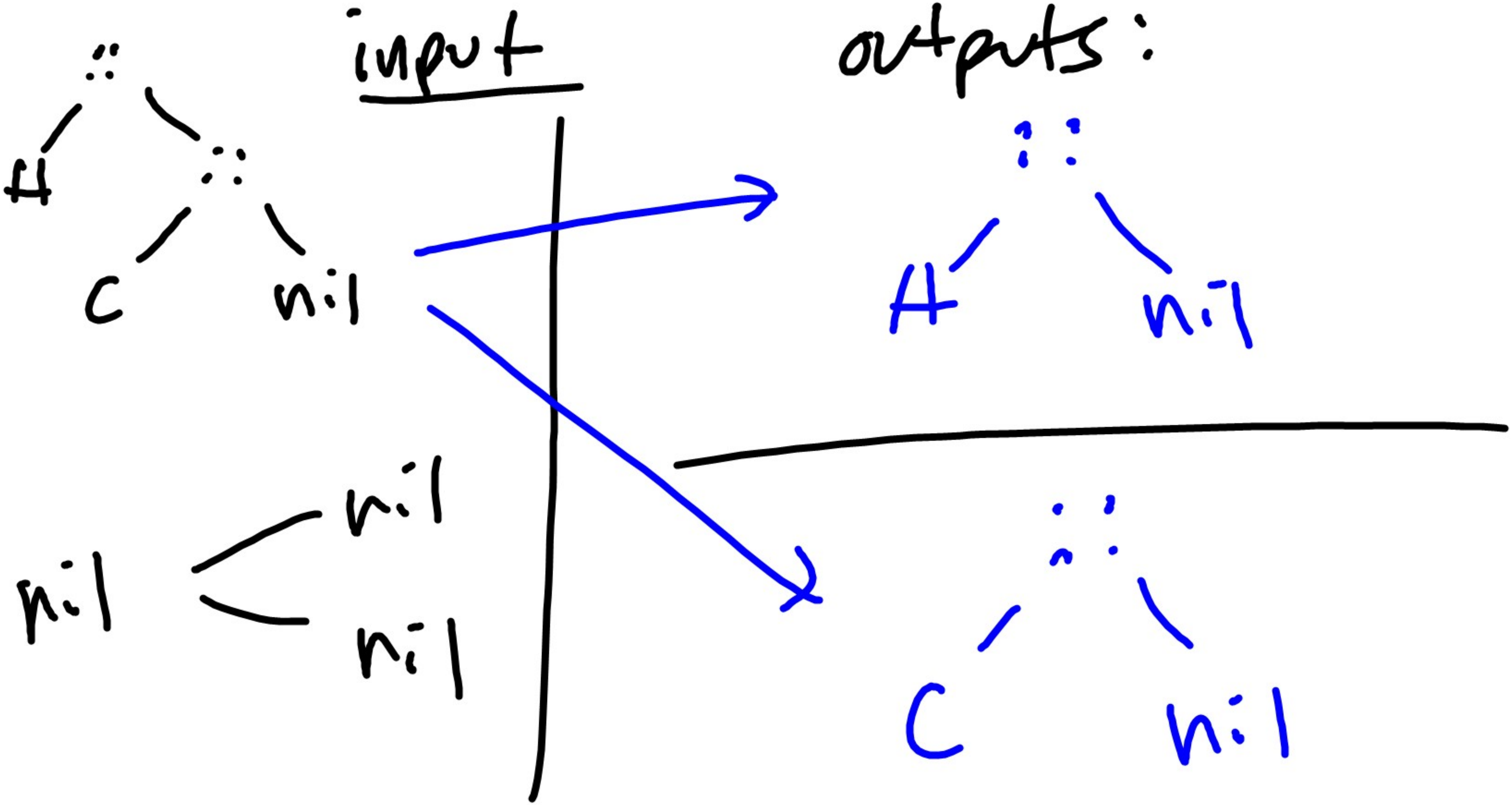
list cell
(cons cell)

$(H :: (S :: (D :: (C :: nil))))$

$\boxed{\begin{array}{c} C :: S \\ Cons(C, S) \end{array}}$

$Cons(H, Cons(D, \ldots$


::
H
::
S
::
C
nil



input
outputs:
::
H
::
C
nil
nil
nil
nil
::
H
nil
::
C
nil

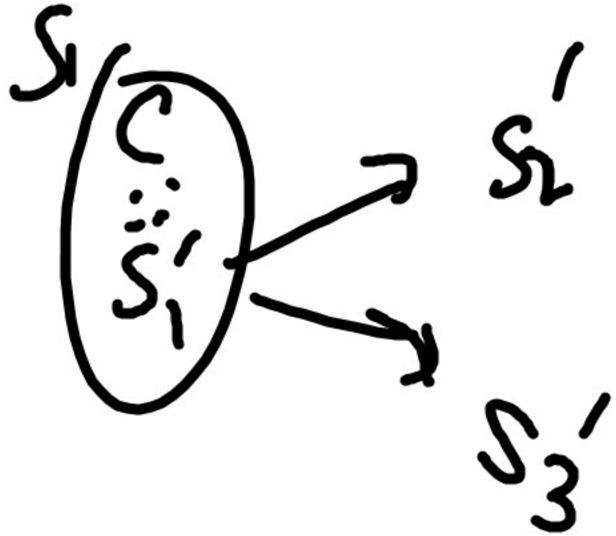
$$
\begin{tikzcd}
 & \mathrm{nil} \\
{[H, C]} \arrow[ur] \arrow[dr] & \\
 & {[H, C]}
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & {[H, C]} \\
{[H, C]} \arrow[ur] \arrow[dr] & \\
 & \mathrm{nil}
\end{tikzcd}
$$

$$\boxed{uns(\underbrace{s_1}_{\text{inp}}, \underbrace{s_2, s_3}_{\text{outputs}})}$$

$$s ::= nil \mid c :: s$$

$$\frac{}{uns(nil, nil, nil)}\ nil$$



$$\frac{uns(s_1', s_2', s_3')}{uns(\underbrace{c :: s_1'}_{s_1}, c :: s_2', s_3')}$$

$$\frac{}{uns(nil, nil, nil)} \; 7$$

$$\frac{uns(s_1, s_2, s_3)}{uns(c :: s_1, c :: s_2, s_3)} \; 9 \qquad \frac{uns(s_1, s_2, s_3)}{uns(c :: s_1, s_2, c :: s_3)} \; 8$$

$$\overline{\mathrm{uns}([\,],[\,],[\,])}\ \mathrm{nil}$$

$$\overline{\mathrm{uns}([D],[D],[\,])}\ L$$

$$\overline{\mathrm{uns}([\underline{S},D],[D],[\underline{S}])}\ R$$

$$\overline{\mathrm{uns}([S,S,D],[S,D],[S])}\ L$$

$$\overline{\mathrm{uns}(H::S::S::D::\mathrm{nil},\ S::D::\mathrm{nil},\ H::S::\mathrm{nil})}\ R$$

$[\,] = \mathrm{nil}$

$$\dfrac{\dfrac{\text{Constructive}}{\text{evidence}}}{\dfrac{\overline{\qquad\qquad} \quad \neg(\text{plus}(n_1, n_2) \Downarrow n_3)}{\text{blah}\cdot\text{blah}}}$$

existential...

THM: $\forall S_1. \exists S_2, S_3$

such that $\text{uns}(S_1, S_2, S_3)$

univ. Quantification

Proof $\approx$ program

IH $\approx$ recursion

$\forall s_1 \; \exists s_2 \; s_3 \quad \text{uns}(s_1, s_2, s_3)$

pf. by ind on s_1

nil

Case nil: $\text{uns}(\text{nil}, \text{nil}, \text{nil})$

recursion

Case $c :: s_1'$:

1. $\text{uns}(s_1', s_2', s_3')$ by IH

s_2 s_3

2. $\text{uns}(\underbrace{c :: s_1'}_{s_1}, c :: s_2', s_3')$

by rule L.

$$f(x) =$$

$$\cdots f(x) \cdots$$

DAY #2 P. of PL.

BNF, $\dfrac{\text{inf rules}}{\text{judgements.}}$, ind proofs...

- syntax — BNF — variables...
  with < subst.
- semantics ~ statics (type sys)
  ~ dynamics (op. sem.)

meta variable

$$n ::= z \mid S(n)$$

x program variable

$$e ::= \lambda x.\, e \mid e_1(e_2)$$

$$\underbrace{\lambda x.(x+1)}_{e}$$

$$e ::= \lambda x.e \mid e_1(e_2) \mid x$$

$$e_1 \longmapsto e_2$$

"e_1 reduces to e_2 in one step"

$$e \Downarrow v$$

"e evaluates to value v"

$$e ::= n \mid s \mid e_1 + e_2 \mid e_1 \wedge e_2$$

$$\tau ::= \mathrm{num} \mid \mathrm{str}$$

$\boxed{e : \tau}$ "expression e has type τ"

$$\frac{}{n : \mathrm{num}}\,\mathrm{num} \qquad \frac{}{s : \mathrm{str}}\,\mathrm{str}$$

$$\frac{e_1 : \mathrm{num} \qquad e_2 : \mathrm{num}}{e_1 + e_2 : \mathrm{num}}$$

$$\frac{e_1 : \mathrm{str} \qquad e_2 : \mathrm{str}}{e_1 \wedge e_2 : \mathrm{str}}$$

$$e ::= \mathrm{let}\ x = e_1\ \mathrm{in}\ e_2$$

$$\mid e_1 + e_2$$

$$\mid \ldots$$

$$\mathrm{let}\ x = 1+2\ \mathrm{in}\ x * x$$