Defining a programming language
Implementing a programming language

Operational semantics
Denotational semantics → Mathematical

semantics → operational because
it gives the step-by-step execution.

— x —

Categories : values, expressions, definitions

values $\subsetneq$ expressions
e.g. in arithmetic 1729 is a value
but $10*10*10+9*9*9$ is not.
The "dynamics" of a language are the
evaluation rules
$10*10*10+9*9*9 \xrightarrow{*} 1729$
evaluation.

<u>nano ML</u>

$exp ::= n \mid true \mid false \mid e_1 + e_2 \mid e_1 * e_2 \mid$
$if\ b\ then\ e_1\ else\ e_2 \mid b_1\ \underline{and}\ b_2 \mid \underline{not}\ b$

NOTATION $e \Downarrow v$

Expression e terminates with value v.

<u>Base</u> $n \Downarrow n$ $\quad true \Downarrow true$

<u>Inductive</u>

$$\frac{e_1 \Downarrow v_1 \qquad e_2 \Downarrow v_2}{e_1 + e_2 \Downarrow v_1 + v_2}$$

$$\frac{e_1 \Downarrow v_1 \qquad e_2 \Downarrow v_2}{e_1 * e_2 \Downarrow v_1 + v_2}$$

← Assume what's above the line holds then what's below the line follows.

$$\frac{b \Downarrow \text{true} \qquad e_1 \Downarrow v_1}{\text{if } b \text{ then } e_1 \text{ else } e_2 \Downarrow v_1} \qquad \frac{b \Downarrow \text{false} \qquad e_2 \Downarrow v_2}{\text{if } b \text{ then } e_1 \text{ else } e_2 \Downarrow v_2}$$

<u>Thm</u> Every expression in nano ML terminates with a value.

<u>Thm</u> If $e \Downarrow v_1$ and also $e \Downarrow v_2$ then $v_1 = v_2$.

---

micro ML ... $|x|$ let $x = e_1$ in e_2

let $x = e_1$ in $\underbrace{e_2}$

x should occur in e_2; (does not have to).

The variable x defined here is only bound to the value of e_1 inside e_2

e_2 is the <u>scope</u> of the binding.

If we look at e_2 in isolation we would think x is unbound.

x is said to be <u>free</u> in e_2

FV : free variables of an expression

$FV(n) = \emptyset$

$FV(e_1 + e_2) = FV(e_1) \cup FV(e_2)$

$FV(x) = \{x\}$

$FV(\text{let } x = e_1 \text{ in } e_2) = FV(e_1) \cup (FV(e_2) \setminus \{x\})$

A variable that is not free is called a <u>bound</u> variable.

$$\int_{-\infty}^{+\infty} e^{-x^2}\,dx = \int_{-\infty}^{\infty} e^{-y^2}\,dy = \sqrt{\pi}$$

$$\forall x\ \exists y.\ \varphi(x,y)$$
$$\equiv \forall u\ \exists v.\ \varphi(u,v)$$

$$\forall \bullet\ \exists \bullet.\ \varphi(\bullet, \bullet)$$

$$FV\left(\begin{array}{l}\text{let } x = 2 \text{ in} \\ \quad \text{let } y = x+1 \text{ in} \\ \qquad x+y\end{array}\right)$$ closed

$$= FV(2) \cup (FV(\underline{\text{let } y = x+1 \text{ in } x+y}) \setminus \{x\})$$
$$= \{x\} \setminus \{x\} = \emptyset$$

$$FV(\text{let } y = x+1 \text{ in } x+y)$$
$$= FV(x+1) \cup \{ FV(x+y) \setminus \{y\})$$
$$= \{x\} \cup \{x\} = \{x\}$$

SUBSTITUTION

let $x = 2$ in
  let $y = x+1$ in
    $x * y$

substitute the expression $x+1$ for y in $x*y$

let $x=2$ in
  $x * (x+1)$

$[e/x]\, e'$ means
replace <u>free</u> occurrences of x in e'
by e

$[x+1/y](y+y) \rightsquigarrow (x+1)+(x+1)$

$[e/x]\, e'$ can easily be defined
by induction on e'

$[e/x]\, x = e$
$[e/x]\,(e_1+e_2) = ([e/x]e_1)+([e/x]e_2)$

~~$[e/x]$ (in let $u=x$~~
$[e/x]$ (let $u=e_1$ in e_2)
  let $u=([e/x]e_1)$ in $([e/x]e_2)$

DANGER "u" may occur free in e
after substitution this u becomes
bound. CAPTURE of free variables.
AVOID it by renaming bound
variables.

$[u+5/z]\,(\text{let } u = x+17 \text{ in } u+z)$

$\text{let } u = x+17 \text{ in } u+(u+5)$

CAPTURED

$$\frac{e_1 \Downarrow v_1 \qquad [v_1/x]\,e_2 \Downarrow v_2}{(\text{let } x = e_1 \text{ in } e_2) \Downarrow v_2}$$

<u>Thm</u> All terms in μML terminate

<u>Thm</u> (DETERMINACY)

MINI-ML

$exp ::= \ldots \mid \text{fun } x \to e \mid e_1 \;\; e_2$

fun $x \to e$ already a value even if e is not

$(\text{fun } x \to e) \Downarrow (\text{fun } x \to e)$

$(e_1 \;\; e_2) \Downarrow ??$

$$\frac{e_1 \Downarrow (\text{fun } x \to e) \qquad e_2 \Downarrow v \qquad [v/x]\,e \Downarrow v'}{(e_1 \;\; e_2) \Downarrow v'}$$

This completes the semantics of a higher-order functional language.