1 Operational Semantics

Definition 1.1 (Normal form) *An expression E is in normal form and irreducible if there is no such E' such that $E \to E'$. Either* ***answer configurations*** *or* ***stuck configurations****.*

Definition 1.2 (Strictness) *An operation is called strict in one of its arguments if it always needs to evaluate that argument. + is strict in both arguments but & is a left-strict operator (non-strict in right).*

1.1 Big-Step/Natural Semantics

- **(Determinacy)** $\forall E \in \mathit{SimpleExp}.\forall n_1, n_2 \in \mathbb{N}.[E \Downarrow n_1 \wedge E \Downarrow n_2 \Rightarrow n_1 = n_2]$.
- **(Totality)** $\forall E \in \mathit{SimpleExp}.\exists n \in \mathbb{N}.[E \Downarrow n]$.(store breaks totality due to stuck configurations)

1.2 Small-Step/Structural Semantics

- **(Determinism)** For all E, E_1, E_2, if $E \to E_1$ and $E \to E_2$, then $E_1 = E_2$
- **(Confluence)** For all E, E_1, E_2, if $E \to^* E_1$ and $E \to^* E_2$, then there exists E' such that $E_1 \to^* E'$ and $E_2 \to^* E'$
- **(Weak Normalisation)** For any expression E_1 there exists a finite sequence of expressions such that for all $i \in [1..k), E_i \to E_{i+1}$
- **(Strong Normalisation)** There are no infinite sequences of expressions such that for all $i, E_i \to E_{i+1}$
- **(Unique Normal Form)** For all E, n_1, n_2, if $E \to^* n_1$ and $E \to^* n_2$ then $n_1 = n_2$

1.3 Many Steps of Evaluation

Definition 1.3 (Reflexive Transitive Closure) $E \to^* E'$ *iff* $E = E'$ *or there is a finite sequence* $E \to E_1 ... \to E_k \to E'$

Theorem 1.4 *For all E and n, $E \Downarrow n$ iff $E \to^* n$*

1.4 Simple Expressions

$$E \in \mathit{SimpleExp} ::= \mathbf{n} \mid E + E \mid E \times E$$

$$\text{(B-NUM)}\ \frac{}{n \Downarrow n} \qquad \text{(B-ADD)}\ \frac{E_1 \Downarrow n_1 \qquad E_2 \Downarrow n_2}{E_1 + E_2 \Downarrow n_3}\ n_3 = n_1 + n_2$$

$$\text{(S-LEFT)}\ \frac{E_1 \to E_1'}{E_1 + E_2 \to E_1' + E_2} \qquad \text{(S-RIGHT)}\ \frac{E \to E'}{n + E \to n + E'} \qquad \text{(S-ADD)}\ \frac{}{n_1 + n_2 \to n_3}\ n_3 = n_2 + n_1$$

1.5 While

$$B \in \mathit{Bool} ::= \mathbf{true} \mid \mathbf{false} \mid E = E \mid E < E \mid ... \mid B\&B \mid \neg B \mid ...$$

$$E \in \mathit{Exp} ::= x \mid \mathbf{n} \mid E + E \mid ...$$

$$C \in \mathit{Com} ::= x := E \mid \text{if } B \text{ then } C \text{ else } C \mid C; C \mid \mathbf{skip} \mid \text{while } B \text{ do } C$$

$$\text{(W-EXP.LEFT)}\ \frac{\langle E_1, s\rangle \to_e \langle E_1', s'\rangle}{\langle E_1 + E_2, s\rangle \to_e \langle E_1' + E_2, s'\rangle} \qquad \text{(W-EXP.RIGHT)}\ \frac{\langle E, s\rangle \to_e \langle E', s'\rangle}{\langle n + E, s\rangle \to_e \langle n + E', s'\rangle}$$

$$\text{(W-EXP.VAR)}\ \frac{}{\langle x, s\rangle \to_e \langle n, s\rangle}\ s(x) = n \qquad \text{(W-EXP.ADD)}\ \frac{}{\langle n_1 + n_2, s\rangle \to_e \langle n_3, s\rangle}\ n_3 = n_1 + n_2$$

$$\text{(W-ASS.EXP)}\ \frac{\langle E, s\rangle \to_e \langle E', s'\rangle}{\langle x := E, s\rangle \to_c \langle x := E', s'\rangle} \qquad \text{(W-ASS.NUM)}\ \frac{}{\langle x := n, s\rangle \to_c \langle \text{skip}, s[x \mapsto n]\rangle}$$

$$\text{(W-SEQ.LEFT)}\ \frac{\langle C_1, s\rangle \to_c \langle C_1', s'\rangle}{\langle C_1; C_2, s\rangle \to_c \langle C_1'; C_2, s'\rangle} \qquad \text{(W-SEQ.SKIP)}\ \frac{}{\langle \text{skip}; C_2, s\rangle \to_c \langle C_2, s\rangle}$$

$$\text{(W-COND.TRUE)}\ \frac{}{\langle \text{if true then } C_1 \text{ else } C_2, s\rangle \to_c \langle C_1, s\rangle}$$

$$\text{(W-COND.FALSE)}\ \frac{}{\langle \text{if false then } C_1 \text{ else } C_2, s\rangle \to_c \langle C_2, s\rangle}$$

$$\text{(W-COND.FALSE)}\ \frac{\langle B, s\rangle \to_b \langle B', s'\rangle}{\langle \text{if } B \text{ then } C_1 \text{ else } C_2, s\rangle \to_c \langle \text{if } B' \text{ then } C_1 \text{ else } C_2, s'\rangle}$$

$$\text{(W-WHILE)}\ \frac{}{\langle \text{while } B \text{ do } C, s\rangle \to_c \langle \text{if } B \text{ then } (C; \text{while } B \text{ do } C) \text{ else skip}, s\rangle}$$

2 Induction

Principle:
For any $P \subseteq \mathbb{N}$:

$$P(0) \land \forall k : \mathbb{N}.[P(k) \to P(k+1)] \longrightarrow \forall n : \mathbb{N}.P(n)$$

Proof Schema:
Base Case:
To Show: P(0)
Inductive Step: Take arbituary k
Inductive Hypothesis: P(k)
To Show: P(k+1)

$$B \in \mathit{Bool} ::= \text{true} \mid \text{false} \mid B\&B \mid \neg B$$

$$\text{(BS1)}\ \frac{}{\text{false } \&B_2 \to \text{false}} \qquad \text{(BS2)}\ \frac{}{\text{true } \&B_2 \to B_2} \qquad \text{(BS3)}\ \frac{B_1 \to B_1'}{B_1\&B_2 \to B_1'\&B_2}$$

$$\text{(BS4)}\ \frac{B \to B'}{\neg B \to \neg B'} \qquad \text{(BS5)}\ \frac{}{\neg\text{true} \to \text{false}} \qquad \text{(BS6)}\ \frac{}{\neg\text{false} \to \text{true}}$$

$$\text{(BL1)}\ \frac{B_1 \Downarrow \text{false}}{B_1\&B_2 \Downarrow \text{false}} \qquad \text{(BL2)}\ \frac{B_1 \Downarrow \text{true} \qquad B_2 \Downarrow b}{B_1\&B_2 \Downarrow b}$$

$$\text{(BL3)}\ \frac{B \Downarrow \text{true}}{\neg B \Downarrow \text{false}} \qquad \text{(BL4)}\ \frac{B \Downarrow \text{false}}{\neg B \Downarrow \text{true}} \ \text{(BL5)}\ \frac{}{b \Downarrow b}$$

2.1 Terms

$$P(true) \land P(false) \land \forall B_1, B_2 \in \mathit{Bool}.[P(B_1) \land P(B_2) \Rightarrow P(B_1\&B_2)] \land \forall B \in \mathit{Bool}.[P(B) \Rightarrow P(\neg B)]$$
$$\Longrightarrow \forall B \in \mathit{Bool}.[P(B)]$$

Split inductive cases into further cases based on step applied

2.2 Derivations

$$\forall B_1, B_2 \in \mathit{Bool}.[Q(B_1, \text{false}) \Rightarrow Q(B_1\&B_2, \text{false})]$$
$$\land \forall B_1, B_2 \in \mathit{Bool}.\forall b \in \mathbb{B}[Q(B_1, \text{true}) \land Q(B_2, b) \Rightarrow Q(B_1\&B_2, b)]$$
$$\land \forall B \in \mathit{Bool}.[Q(B, \text{true}) \Rightarrow Q(\neg B, \text{false})] \land \forall B \in \mathit{Bool}.[Q(B, \text{false}) \Rightarrow Q(\neg B, \text{true})] \land \forall b \in \mathbb{B}.[Q(b, b)]$$
$$\Longrightarrow \forall B \in \mathit{Bool}.\forall b \in \mathbb{B}.[B \Downarrow b \Rightarrow Q(B, b)]$$

2.3 Many Steps

$$R^*(a, a') \triangleq a = a' \lor \exists a'' \in A.[R(a, a'') \land R^*(a'', a')]$$
$$\forall a \in A.Q(a, a) \land \forall a, a', a'' \in A.[R(a, a'') \land Q(a'', a') \Rightarrow Q(a, a')] \Longrightarrow \forall a, a' \in A.[R^*(a, a') \Rightarrow Q(a, a')]$$
$$R^+(a, a') \triangleq R(a, a') \lor \exists a'' \in A.[R(a, a'') \land R^+(a'', a')]$$
$$\forall a \in A.[R(a, a') \Rightarrow Q(a, a')] \land \forall a, a', a'' \in A.[R(a, a'') \land Q(a'', a') \Rightarrow Q(a, a')] \Longrightarrow \forall a, a' \in A.[R^+(a, a') \Rightarrow Q(a, a')]$$
$$\forall E \in S.[\forall E_2 \in S.[(E + E_2) \to^* (E + E_2)]] \land$$
$$\forall E_1, E_1', E_1'' \in S.[E_1 \to E_1'' \land \forall E_2 \in S.[E_1'' + E_2 \to^* E_1' + E_2]] \Rightarrow \forall E_2 \in S.[E_1 + E_2 \to^* E_1' + E_2]$$
$$\Longrightarrow \forall E_1, E_1', E_2 \in S.[E_1 \to^* E_1' \Rightarrow (E_1 + E_2) \to^* (E_1' + E_2)]$$

3 Machines

3.1 Register Machine

Instructions $L_0 : R_0^+ \to L_1 \quad L_1 : R_0^- \to L_0, L_2 (R_0 > 0 ? L_0 : L_2) \quad L_2 : \text{HALT}$
Configuration $c = (l, r_0, ..., r_n)$ where l is the label and r_i is the contents of R_i, initally with $l = 0$

Definition 3.1 (Computable Functions) *Partial function $f \in \mathbb{N}^n \to \mathbb{N}$ is (register machine) computable if there is a register machine M with at least $n+1$ registers $R_0, R_1, ..., R_n$ (and maybe more) such that for all $(x_1, ..., x_n) \in \mathbb{N}^n$ and all $y \in \mathbb{N}$, the computation of M starting with $R_0 = 0, R_1 = x_1, ..., R_n = x_n$ and all other registers set to 0, halts with $R_0 = y$ if and only if $f(x_1, ..., x_n) = y$.*

Definition 3.2 (Numerical Coding of Pairs) $\begin{cases} \langle\langle x, y \rangle\rangle \triangleq 2^x(2y+1) = 0b\boldsymbol{y}10...0(\boldsymbol{x} \textit{ number of 0s}) \\ \langle x, y \rangle \triangleq 2^x(2y+1) - 1 \end{cases}$

Definition 3.3 (Numerical Coding of Lists) $\begin{cases} \ulcorner [] \urcorner \triangleq 0 \\ \ulcorner x :: l \urcorner \triangleq \langle\langle x, \ulcorner l \urcorner \rangle\rangle \triangleq 2^x(2 \cdot \ulcorner l \urcorner + 1) \end{cases}$

$\ulcorner [x_1, x_2, ..., x_n] \urcorner = 10.._{x_n}..010.._{x_{n-1}}..01...10.._{x_1}..0$

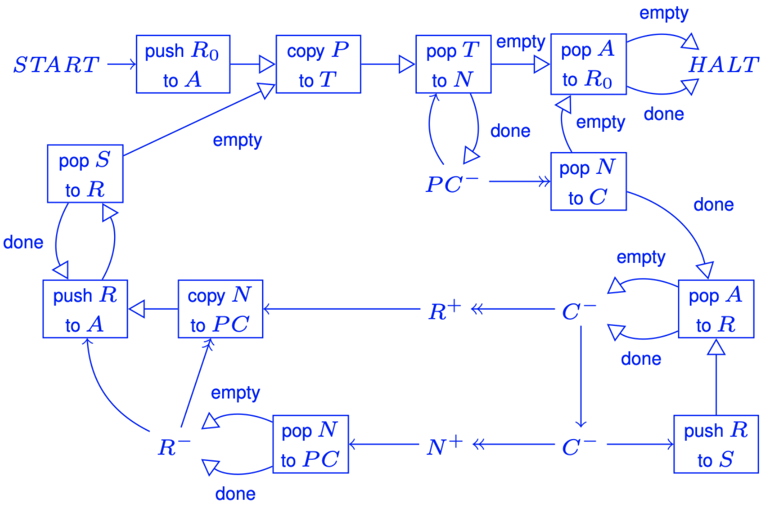
Definition 3.4 (Numerical Coding of Programs) $\begin{cases} \ulcorner R_i^+ \to L_j \urcorner \triangleq \langle\langle 2i, j \rangle\rangle \\ \ulcorner R_i^- \to L_j, L_k \urcorner \triangleq \langle\langle 2i+1, \langle j, k \rangle \rangle\rangle \\ \ulcorner HALT \urcorner \triangleq 0 \end{cases}$

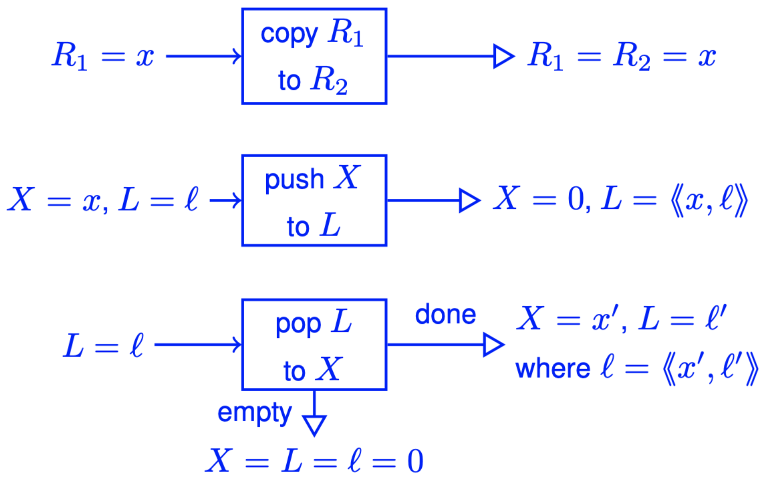
$1 \triangleq \langle\langle 0,0 \rangle\rangle \triangleq R_0^+ \to L_0 \quad 2 \triangleq \langle\langle 1,0 \rangle\rangle \triangleq R_0^- \to L_0, L_0 \quad 3 \triangleq \langle\langle 0,1 \rangle\rangle \triangleq R_0^+ \to L_1 \quad 4 \triangleq \langle\langle 2,0 \rangle\rangle \triangleq R_1^+ \to L_0 \quad 5 \triangleq \langle\langle 0,2 \rangle\rangle \triangleq R_0^+ \to L_2$

Definition 3.5 (Universal Register Machine) *Starts with $R_0 = 0, R_1 = e$ (code of program) and $R_2 = a$ (list of arguments) with all other registers zeroed.*

- *decodes e as a RM program P*
- *decodes a as a list of register values $a_1, ..., a_n$*
- *computes P starting with $R_0 = 0, R_1 = a_1, ..., R_n = a_n$ and all other registers in P set to 0*

*R_1 =**P**rogram code, R_2 =List of RM **A**rguments, R_3 =**PC**, R_4=**N**ext inst label, R_5 =**C**urrent inst code, R_6 =Value of current **R**egister, R_7&R_8=**S**cra**T**ch register*





Theorem 3.6 (Halting Problem) *RM H decides the halting problem if started with $R_0 = 0, R_1 = e$ (code of program) and $R_2 = a$ (list of arguments) with all other registers zeroed, always halts with $R_0 =$ 0 or 1, with $R_0 = 1$ iff the RM with index e eventually halts when started with a.*
Construct H' by copying R_1 to R_2 before running H. Construct C by modifying H' so that C halts if $R_0 = 0$ and loops forever if $R_0 > 0$. Let c be the code of C.
C started with $R_1 = c$ halts iff H' started with $R_1 = c$ halts with $R_0 = 0$ iff H started with $R_1 = c, R_2 = \ulcorner [c] \urcorner$ halts with $R_0 = 0$ iff C started with $R_1 = c$ does not halt.

Definition 3.7 (Decidable) *$S \subseteq \mathbb{N}$ is RM decidable iff there is a RM started with $R_0 = 0, R_1 = x$ and all other registers zeroed halts with $R_0 =$ 0 or 1 and $R_0 = 1$ iff $x \in S$*

Definition 3.8 (Semidecidable) *There is a TM that always halts if the input value belongs to the set or run forever otherwise*

3.2 Turing Machine

$M = (Q, \Sigma, s, \delta)$ where Q is a finite set of machine states, Σ is a set of tape symbols containing the distinguished blank symbol $\sqcup$, a initial state $s \in Q$, and a partial transition function $\delta \in (Q \times \Sigma) \rightharpoonup (Q \times \Sigma \times \{L, R\})$
Configuration (q, w, u) comprising current state $q \in Q$, finite possibly empty string $w \in \Sigma^*$ of symbols left of tape head, and finite possibly empty string $u \in \Sigma^*$ of symbols under and right of tape head.
Initial configuration of (s, ϵ, u) with ϵ denoting the empty string.

$$\frac{\text{first(u)=(a,u')} \qquad \delta\text{(q,a)=(q',a',L)} \qquad \text{last(w)=(b,w')}}{\text{(q,w,u)}\to_M\text{(q',w',ba'u')}} \qquad \frac{\text{first(u)=(a,u')} \qquad \delta\text{(q,a)=(q',a',R)}}{\text{(q,w,u)}\to_M\text{(q',wa',u')}}$$

Definition 3.9 (Tape Encoding of Lists) *A tape over $\Sigma = \{\sqcup, 0, 1\}$ where precisely two cells contain 0 and the only cells containing 1 occur between these two*
$... \sqcup \sqcup 01.._{n_1}..1 \sqcup 1.._{n_2}..1 \sqcup ... \sqcup 1.._{n_k}..10 \sqcup \sqcup$

Definition 3.10 (Computable) *$f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ is Turing computable iff there exists a TM M starting on leftmost 0 on tape coding $[x_1, ..., x_n]$, M halts iff $f(x_1, ..., x_n)\downarrow$ and the final tape codes a list whose first element is y where $f(x_1, ..., x_n) = y$.*

Theorem 3.11 (Church-Turing Thesis) *Every algorithm can be realised as a Turing machine*

3.3 Lambda Calculus

Definition 3.12 (Free Variables) $FV(x) = \{x\} \quad FV(\lambda x.M) = FV(M) \backslash \{x\} \quad FV(MN) = FV(M) \cup FV(N)$

Definition 3.13 (α-equivalence) *$M =_\alpha N$ iff one can be obtained from another by renaming bound variables (must have same set of free variables)*

Definition 3.14 (Substitution)

$$x[M/y] = \begin{cases} M & x = y \\ x & x \neq y \end{cases} \quad (\lambda x.N)[M/y] = \begin{cases} \lambda x.N & x = y \\ \lambda z.N[z/x][M/y] & x \neq y \end{cases} \quad (M_1 M_2)[M/y] = (M_1[M/y])(M_2[M/y])$$

Definition 3.15 (β-reduction)

$$\frac{}{(\lambda x.M)N \to_\beta M[N/x]} \qquad \frac{M =_\alpha M' \qquad M' \to_\beta N' \qquad N' =_\alpha N}{M \to_\beta N}$$

$$\frac{M \to_\beta M'}{\lambda x.M \to_\beta \lambda x.M'} \qquad \frac{M \to_\beta M'}{MN \to_\beta M'N} \qquad \frac{N \to_\beta N'}{MN \to_\beta MN'}$$

Definition 3.16 (Multi-Step β-reduction)

$$Reflexivity\frac{M =_\alpha M'}{M \to_\beta^* M'} \quad Transitivity\frac{M \to_\beta M'' \qquad M'' \to_\beta^* M'}{M \to_\beta^* M'}$$

Theorem 3.17 (Church-Rosser Confluence)

$$\forall M, M_1, M_2.[M \to_\beta^* M_1 \land M \to_\beta^* M_2 \Rightarrow \exists M'.M_1 \to_\beta^* M' \land M_2 \to_\beta^* M']$$

Theorem 3.18 (Uniqueness of Normal Form)

$$\forall M, N_1, N_2.[M \to_\beta^* N_1 \land M \to_\beta^* N_2 \land is\_in\_nf(N_1) \land is\_in\_nf(N_2) \Rightarrow N_1 =_\alpha N_2]$$

Definition 3.19 (β-equivalence)

$$M_1 =_\beta M_2 \iff \exists M'.M_1 \to_\beta^* M' \land M_2 \to_\beta^* M'$$

Definition 3.20 (Reduction Strategies)

- *For $E = (\lambda x.M)N$, M and N is inside E.*
- ***Outermost*** *redex if no redex outside it.* ***Innermost*** *redex if no redex inside it.*
- ***Normal Order*** *leftmost outermost, always reduces to normal form if it exists*
- ***Call by Name*** *leftmost outermost & does not reduce inside λ-abstraction, pass unevaluated parameters into body which are evaluated on each use, does not always terminate*
- ***Call by Value*** *leftmost innermost & does not reduce inside λ-abstraction, evaluate function parameters before passing them into body, terminates less often than Call by Name*

Definition 3.21 (Definability) *$f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ is λ-definable iff there exists a closed λ-term M where $f(x_1, ..., x_n) = y$ iff $M\underline{x_1}...\underline{x_n} =_\beta \underline{y}$ and $f(x_1, ..., x_n) \uparrow$ iff $M\underline{x_1}...\underline{x_n}$ has no normal form*

Definition 3.22 (Church Encoding)

$$\underline{n} \triangleq \lambda f.\lambda x.f(..._n..(f(x))...) \quad plus \equiv \lambda m.\lambda n.\lambda f.\lambda x.mf(nfx) \quad mult \equiv \lambda m.\lambda n.\lambda f.m(nf)$$

$$\underline{m^n} \triangleq \lambda m.\lambda n.nm \quad ifz \triangleq \text{if (m=0) then } x_1 \text{ else } x_2 \triangleq \lambda m.\lambda x_1.\lambda x_2.m(\lambda z.x_2)x_1$$

$$pair \triangleq \lambda v_1, v_2.(\lambda p.pv_1v_2) \quad fst \triangleq \lambda q.q(\lambda w_1 w_2.w_1) \quad snd \triangleq \lambda q.q(\lambda w_1 w_2.w_2)$$

$$succ \triangleq \lambda nfx.f(nfx) \quad pred \triangleq \lambda nfx.n(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u) \quad minus \triangleq \lambda mn.(n\ pred)\ m$$

$$pred \triangleq \lambda n.fst(nf(pair\ \underline{0}\ \underline{0})) \text{ where } f \triangleq \lambda p.pair\ (snd\ p)\ (succ\ (snd\ p))$$

Definition 3.23 (Combinators) *Closed lambda terms, no free variables*

$$\begin{array}{lll} I \triangleq \lambda x.x & K \triangleq \lambda xy.x & S \triangleq \lambda xyz.xz(yz) \\ T \triangleq \lambda xy.yx & C \triangleq \lambda xyz.xzy & V \triangleq \lambda xyz.zxy \\ B \triangleq \lambda xyz.x(yz) & B' \triangleq \lambda xyz.y(xz) & W \triangleq \lambda xy.xyy \end{array}$$

$Y \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, *after one step* $Yf \to_\beta f(Yf)$.

Example: factorial = $\lambda f.\lambda x.$ if $x == 1$ then 1 else $x * f(x-1)$. Yfactorial n = Y(factorial)(n).