Daisy Summary

These figures and indices are assembled from the the full manual:

1. Language summary
   A. Daisy expressions
   B. Composite forms, binding, conditional, surface assignment
   C. Daisy grammar (plus surface assignment)
2. Daisy semantics
   A. Evaluation
   B. Application
   C. Other evaluation laws
   D. Indeterminate lists
3. Environments
4. Closure objects
5. Operations
   A. by group
   B. alphabetical
   C. table of arithmetic operations

6 Evaluation errons and syntax errons

1A. Daisy Expressions

Expressions, E

numerals, N

literals, I

`"` —*characters*— `"`	(literal quotation)
`^` E	(value quotation)
`(` E `)`	(parentheses)
E_0 `:` E_1	(application expression)
`\` X `.` E	(function expression)
`[` `]`	(Nil)
`[` $E_0 \cdots E_n$ `]`	(list value)
`[` $E_0 \cdots E_n$ `!` E_{n+1} `]`	(list value)
`[` $E_0 \cdots E_n$ `*` `]`	(list value)
`<` `>`	(Nil)
`<` $E_0 \cdots E_n$ `>`	(list expression)
`<` $E_0 \cdots E_n$ `!` E_{n+1} `>`	(list expression)
`<` $E_0 \cdots E_n$ `*` `>`	(list expression)
`{` `}`	(Nil)
`{` $E_0 \cdots E_n$ `}`	(list expression)
`{` $E_0 \cdots E_n$ `!` E_{n+1} `}`	(list expression)
`{` $E_0 \cdots E_n$ `*` `}`	(list expression)

Formal Arguments, X

literals

`[` `]`

`[` $X_0 \cdots X_n$ `]`

`[` `[` $X_0 \cdots X_n$ `!` X_{n+1} `]`

1B. Composite Forms

Binding Forms

`let:[` X E_0 E_1 `]`
`ret:[` X E_0 E_1 `]`
`fix:[` X `!` E_1 `]`

Conditional Form

`if:<` E_0 E'_0 $\cdots$ E_n E'_n E_{n+1} `>`

Surface Assignment

literal `=` E

1C. A Daisy Grammar (plus surface assignment)

Atomic Expressions

EXP ::= *NUMERAL* | *LITERAL*

List Expressions

EXP ::= `<` *ELTS* `>` | `{` *ELTS* `}` | `[` *ELTS* `]`
ELTS ::= —*nothing*— | *EXP* `!` *EXP* | *EXP* `*` | *EXP ELTS*

Application Expression

EXP ::= *EXP* `:` *EXP*

Function Expression

EXP ::= `\` *ARG* `.` *EXP*
ARG ::= *LITERAL* | `[` *ARGS* `]`
ARGS ::= —*nothing*— | *ARG ARGS* | *ARG* `!` *ARG*

Quotation Expressions

EXP ::= `"`—*characters*—`"`
EXP ::= `^`Exp

Parenthesized Expression

EXP ::= `(`*EXP*`)`

Binding Forms

`let:[`*ARG EXP EXP*`]`
`rec:[`*ARG EXP EXP*`]`
`fix:[`*ARG* `!` *EXP*`]`

Conditional Form

`if:<`*EXP Exp* ··· *EXP*`>`

Surface Assignment

LITERAL `=` *EXP*

2A. Evaluation

$\mathcal{V}_\rho[\![E]\!]$ is the value of expression E in environment ρ. Variables E stand for expressions, I for literal identifiers, and X for function arguments. Lower case variables stand for values.

$$\mathcal{V}_\rho[\![N]\!] = N$$

$$\mathcal{V}_\rho[\![I]\!] = \begin{cases} v, & \text{if } v \text{ is assigned to } I; \\ \rho(I), & \text{otherwise.} \end{cases}$$

$$\mathcal{V}_\rho[\![\texttt{add}]\!] = \texttt{add}$$

$$\mathcal{V}_\rho[\![\texttt{"}\text{—}\textit{characters}\text{—}\texttt{"}]\!] = \text{the literal } \text{—}\textit{characters}\text{—}$$

$$\mathcal{V}_\rho[\![\texttt{\^{}}E]\!] = E$$

$$\mathcal{V}_\rho[\![\texttt{(}E\texttt{)}]\!] = \mathcal{V}_\rho[\![E]\!]$$

$$\mathcal{V}_\rho[\![\backslash X \,.\, E]\!] = {}_\rho\backslash X \,.\, E$$

$$\mathcal{V}_\rho[\![E_0 : E_1]\!] = \mathcal{A}_\rho \left(\mathcal{V}_\rho[\![E_0]\!]\right) \left(\mathcal{V}_\rho[\![E_1]\!]\right)$$

$$\mathcal{V}_\rho[\![\texttt{[ ]}]\!] = \texttt{[ ]}$$

$$\mathcal{V}_\rho[\![\texttt{[} E_0 \cdots E_n \texttt{]}]\!] = \texttt{[} E_0 \cdots E_n \texttt{]}$$

$$\mathcal{V}_\rho[\![\texttt{[} E_0 \cdots E_n \texttt{!} E_{n+1} \texttt{]}]\!] = \texttt{[} E_0 \cdots E_n \texttt{!} E_{n+1} \texttt{]}$$

$$\mathcal{V}_\rho[\![\texttt{[} E_0 \cdots E_n \texttt{*]}]\!] = \texttt{[} E_0 \cdots E_n \texttt{*]}$$

$$\mathcal{V}_\rho[\![\texttt{< >}]\!] = \texttt{[ ]}$$

$$\mathcal{V}_\rho[\![\texttt{<} E_0 \cdots E_n \texttt{>}]\!] = \texttt{[} \mathcal{V}_\rho[\![E_0]\!] \cdots \mathcal{V}_\rho[\![E_n]\!] \texttt{]}$$

$$\mathcal{V}_\rho[\![\texttt{<} E_0 \cdots E_n \texttt{!} E_{n+1} \texttt{>}]\!] = \texttt{[} \mathcal{V}_\rho[\![E_0]\!] \cdots \mathcal{V}_\rho[\![E_n]\!] \texttt{!} \mathcal{V}_\rho[\![E_{n+1}]\!] \texttt{]}$$

$$\mathcal{V}_\rho[\![\texttt{<} E_0 \cdots E_n \texttt{*>}]\!] = \texttt{[} \mathcal{V}_\rho[\![E_0]\!] \cdots \mathcal{V}_\rho[\![E_n]\!] \texttt{*]}$$

2B. Application

$\mathcal{A}_\rho\, u\, v$ interprets the application of value u to value v in environment ρ. Variables E stand for expressions, X for formal arguments, N for numerals.

$$\mathcal{A}_\rho \,\mathsf{add}\,[\,u\ \ v\,] \;=\; u+v$$

$$\mathcal{A}_\rho\, N\,[\,v_0\ \ v_1\ \ \cdots\,] \;=\; v_N$$

$$\mathcal{A}_\rho\,\left({}_{\rho'}\backslash X\,.\,E\right) v \;=\; \mathcal{V}_{\rho''}[\![\,E\,]\!] \text{ where } \rho'' = \rho'\left[\begin{smallmatrix} v \\ X \end{smallmatrix}\right]$$

$$\mathcal{A}_\rho \,\mathsf{val}\, v \;=\; \mathcal{V}_\rho[\![\,v\,]\!]$$

$$\mathcal{A}_\rho \,\mathsf{let}\,[\,X\ \ E\ \ E'\,] \;=\; \mathcal{V}_{\rho'}[\![\,E'\,]\!] \text{ where } \begin{cases} \rho' \;=\; \rho\left[\begin{smallmatrix} v \\ X \end{smallmatrix}\right] \\ v \;=\; \mathcal{V}_\rho[\![\,E_0\,]\!] \end{cases}$$

$$\mathcal{A}_\rho \,\mathsf{rec}\,[\,X\ \ E\ \ E'\,] \;=\; \mathcal{V}_{\rho'}[\![\,E'\,]\!] \text{ where } \begin{cases} \rho' \;=\; \rho\left[\begin{smallmatrix} v \\ X \end{smallmatrix}\right] \\ v \;=\; \mathcal{V}_{\rho'}[\![\,E_0\,]\!] \end{cases}$$

$$\mathcal{A}_\rho \,\mathsf{fix}\,[\,X\ \texttt{!}\ E\,] \;=\; v \text{ where } \begin{cases} \rho' \;=\; \rho\left[\begin{smallmatrix} v \\ X \end{smallmatrix}\right] \\ v \;=\; \mathcal{V}_{\rho'}[\![\,E_0\,]\!] \end{cases}$$

$$\mathcal{A}_\rho \,\mathsf{if}\,[\,u_0\ \ v_0\ \cdots\ u_n\ \ v_n\ \ w\,] \;=\; \begin{cases} v_i, & \text{if } u_i \neq \texttt{[]} \text{ and } u_0 \ldots u_{i-1} \;=\; \texttt{[]} \\ w, & \text{if } u_0 \ldots u_n \;=\; \texttt{[]} \end{cases}$$

$$\mathcal{A}_\rho\,[\,]\,(v) \;=\; [\,]$$

$$\mathcal{A}_\rho\,[\,v\ \texttt{!}\ v'\,]\,[\,[\,u_0\ \texttt{!}\ u_0'\,]\ \cdots\,] \;=\; [\,w\ \texttt{!}\ w'\,] \text{ where } \begin{cases} w = & \mathcal{A}_\rho\, v\,[\,u_0\ \cdots\,] \\ w' = & \mathcal{A}_\rho\, v'\,[\,u_0'\ \cdots\,] \end{cases}$$

$$\mathcal{A}_\rho\,[\,v\ \texttt{*}\,]\,[\,[\,]\ \cdots\,] \;=\; [\,]$$

$$\mathcal{A}_\rho\,[\,v\ \texttt{*}\,]\,[\,[\,u_0\ \texttt{!}\ u_0'\,]\ \cdots\,] \;=\; [\,w\ \texttt{!}\ w'\,] \text{ where } \begin{cases} w = & \mathcal{A}_\rho\, v\,[\,u_0\ \cdots\,] \\ w' = & \mathcal{A}_\rho\,[\,v\ \texttt{*}\,]\,[\,u_0'\ \cdots\,] \end{cases}$$

2C. Other Values

$\mathcal{A}_\rho \perp v = \perp$

$\mathcal{A}_\rho v \perp = \perp$

$\mathcal{A}_\rho \mathit{erron} v = \mathit{erron}$

$\mathcal{A}_\rho [\![I]\!] v = \mathit{error}$

$\mathcal{A}_\rho [\![E_0 : E_1]\!] v = \mathit{error}$

$\mathcal{A}_\rho [\![\backslash X . E]\!] v = \mathcal{V}_\rho [\![E]\!]$ where $\rho = \phi \begin{bmatrix} v \\ X \end{bmatrix}$

$\mathcal{V}_\rho \perp = \perp$

$\mathcal{V}_\rho \mathit{erron} = \mathit{erron}$

$\mathcal{V}_\rho ({}_{\rho'}\backslash X . E) = \mathit{illformed\ closure}$

2D. Indeterminate Lists

The indices $0', 1', \ldots, n'$, $n \geq 0$, *are a rearrangement of* $0, 1, \ldots, n$. *Variables* $A, M, \Omega, E_0, E_1, \ldots$ *stand for expressions whose values are* $a, m, \perp, e_0, e_1, \ldots$, *respectively. Assume* a *is a non-list,* m *is an erron, and that each of the* e_i*s is valid.*

$\mathcal{V}_\rho[\![\texttt{\{ \}}]\!] = \texttt{[ ]}$

$\mathcal{V}_\rho[\![\texttt{\{} E_0 \cdots E_n \texttt{\}}]\!] \stackrel{\triangleright}{=} \texttt{[} e_{0'} \cdots e_{n'} \texttt{]}$

$\mathcal{V}_\rho[\![\texttt{\{} E_0 \texttt{ ! \{} E_1 \cdots E_n \texttt{\}\}}]\!] \stackrel{\triangleright}{=} \texttt{[} e_{0'} \cdots e_{n'} \texttt{]}$

$\mathcal{V}_\rho[\![\texttt{\{}\Omega\ E_0 \cdots E_n \texttt{\}}]\!] \stackrel{\triangleright}{=} \texttt{[} e_{0'} \cdots e_{n'} \texttt{ ! } \perp \texttt{]}$

$\mathcal{V}_\rho[\![\texttt{\{} M\ E_0 \cdots E_n \texttt{\}}]\!] \stackrel{\triangleright}{=} \texttt{[} e_{0'} \cdots e_{n'}\ m \texttt{]}$

$\mathcal{V}_\rho[\![\texttt{\{} E_0 \texttt{ ! \{} E_1 \texttt{ ! } E_2 \texttt{\}\}}]\!] \stackrel{\triangleright}{=} \mathcal{V}_\rho[\![\texttt{\{} E_0\ E_1 \texttt{ ! } E_2 \texttt{\}}]\!]$

$$\mathcal{V}_\rho[\![\texttt{\{} E_0 \texttt{ ! <} E_1 \texttt{ ! } E_2 \texttt{>\}}]\!] \stackrel{\triangleright}{=} \begin{cases} \texttt{[} e_0 \texttt{ ! } \mathcal{V}_\rho[\![\texttt{<} E_1 \texttt{ ! } E_2 \texttt{>}]\!] \texttt{]} \\ \texttt{[} e_1 \texttt{ ! } \mathcal{V}_\rho[\![\texttt{\{} E_0 \texttt{ ! } E_2 \texttt{\}}]\!] \texttt{]} \end{cases}$$

$\mathcal{V}_\rho[\![\texttt{\{} E_0 \cdots E_n\ E_{n+1} \texttt{ *\}}]\!] \stackrel{\triangleright}{=} \texttt{[} e_{0'} \cdots e_{n'}\ e_{n+1} \texttt{ *]}$

$\mathcal{V}_\rho[\![\texttt{\{} E_0 \cdots E_n \texttt{ ! } A \texttt{\}}]\!] \stackrel{\triangleright}{=} \texttt{[} e_{0'} \cdots e_{n'} \texttt{ ! } a \texttt{]}$

3. Environments

Environments ρ associate values with literal identifiers. $\rho(\,I\,)$ is I's binding in ρ. Environment extension is specified as

$$\rho\Big[\begin{smallmatrix} v \\ I \end{smallmatrix}\Big](J) \;=\; \begin{cases} v, & \text{if } I = J \\ \rho(J), & \text{if } I \neq J \end{cases}$$

$$\rho\Big[\begin{smallmatrix} \texttt{[}u\texttt{ ! }v\texttt{]} \\ \texttt{[}X\texttt{ ! }Y\texttt{]} \end{smallmatrix}\Big](J) \;=\; \rho\Big[\begin{smallmatrix} v \\ Y \end{smallmatrix}\Big]\Big[\begin{smallmatrix} v \\ X \end{smallmatrix}\Big](J)$$

$$\rho\Big[\begin{smallmatrix} v \\ \texttt{[]} \end{smallmatrix}\Big](J) \;=\; \rho(J)$$

Environments accept undefined bindings,

$$\rho\Big[\begin{smallmatrix} \bot \\ I \end{smallmatrix}\Big](J) = \begin{cases} \bot, & \text{if } I = J \\ \rho(J), & \text{if } I \neq J \end{cases}$$

The void environment, $\star$ binds every literal to an unbound-identifier erron,

$$\star(\,I\,) = \texttt{|ubi:}I\texttt{|}$$

4. Closure Objects

The closure of a function expression $\backslash\, X\ .\ E$ is denoted ${}_{\rho}\backslash X\,.E$, and appears in output as $\texttt{\textbackslash?\textbackslash}X\ .\ E$.

Daisy Operations
(by group)

Types in descriptors		
	$\mathcal{B}$ – `T` *or* `[]`	
$\mathcal{N}$ – a numeral	$\mathcal{L}$ – a list	$\mathcal{C}$ – a character
$\mathcal{I}$ – a literal	$\mathcal{A}$ – an atom	$\mathcal{E}$ – an expression
	$\mathcal{V}$ – a value	

Arithmetic Tests

`zero?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for Zero*
`one?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for One*
`neg?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for negative*
`pos?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for positive*
`lt?`: `[`$\mathcal{N}_1$ $\mathcal{N}_2$`]` $\longrightarrow$ $\mathcal{B}$ . . . . . *Less-than*
`le?`: `[`$\mathcal{N}_1$ $\mathcal{N}_2$`]` $\longrightarrow$ $\mathcal{B}$ . . . . . *At-most*
`eq?`: `[`$\mathcal{N}_1$ $\mathcal{N}_2$`]` $\longrightarrow$ $\mathcal{B}$ . . . . . *Numeric equality*
`ne?`: `[`$\mathcal{N}_1$ $\mathcal{N}_2$`]` $\longrightarrow$ $\mathcal{B}$ . . . . . *Numeric inequality*
`ge?`: `[`$\mathcal{N}_1$ $\mathcal{N}_2$`]` $\longrightarrow$ $\mathcal{B}$ . . . . . *At-least*
`gt?`: `[`$\mathcal{N}_1$ $\mathcal{N}_2$`]` $\longrightarrow$ $\mathcal{B}$ . . . . . *Greater-than*

Unary Arithmetic

`neg`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Negate*
`sgn`: $\mathcal{N} \longrightarrow$ `-1` or `1` . . . . . *Sign projection*
`inc`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Increment*
`dcr`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Decrement*
`inv`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Invert*

Binary Arithmetic

`add:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Add*
`sub:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Subtract*
`mpy:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Multiply*
`div:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Divide*
`rem:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Remainder*

Binary Logic

`or:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Logical-or*
`and:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Logical-and*
`xor:` $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Exclusive-or*

Reference Tests

`nil?:` $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for Nil*
`isNML?:` $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a numeral*
`isLtrl?:` $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a literal*
`isAtm?:` $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for an Atom*
`isLST?:` $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a list*
`same?:` $[\mathcal{U}\ \mathcal{V}_1 \cdots \mathcal{V}_n] \longrightarrow \mathcal{B}$ . . . . . *Reference equality*

List Processing

`head:` $[\mathcal{V}_1\ \texttt{!}\ \mathcal{V}_2] \longrightarrow \mathcal{V}_1$ . . . . . *Head of a list*
`tail:` $[\mathcal{V}_1\ \texttt{!}\ \mathcal{V}_2] \longrightarrow \mathcal{V}_1$ . . . . . *Tail of a list*
`cons:` $[\mathcal{V}_1\ \mathcal{V}_2] \longrightarrow [\mathcal{V}_1\ \texttt{!}\ \mathcal{V}_2]$ . . . . . *List constructor*
`frons:` $[\mathcal{V}\ \mathcal{V}'] \longrightarrow \mathcal{L}$ . . . . . *Multiset constructor*
`any?:` $[\mathcal{V}_0\ \mathcal{V}_1 \cdots] \longrightarrow [\mathcal{V}_i\ \mathcal{V}_{i+1} \cdots]$ *Locate a non-Nil element*
`all?:` $[\mathcal{V}_0\ \mathcal{V}_1 \cdots] \longrightarrow \mathcal{B}$ . . . . . *Test for no null elements*
`in?:` $[\mathcal{U}\ [\mathcal{V}_0\ \mathcal{V}_1 \cdots]] \longrightarrow \mathcal{B}$ . . . . . *Membership Test*
`if:` $[\mathcal{T}_0\ \mathcal{V}_0\ \mathcal{T}_1\ \mathcal{V}_1 \cdots] \longrightarrow \mathcal{V}_i$ . . . . . *Conditional operation*

Object Manipulation

`TagOf:` $\mathcal{V} \longrightarrow \mathcal{N}$ . . . . . *Numeric value of a tag*

Tag Tests

`isDCT?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a directive*
`isNML?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a numeral*
`isLST?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a list-object*
`isAPL?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a application-object*
`isFTN?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a function-object*
`isIDE?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a identifier-object*
`isERR?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for an identifier-object*

Tag Coercion

`asDCT`: $\mathcal{V} \longrightarrow \mathcal{D}$ . . . . . *Cite as a directive*
`asNML`: $\mathcal{V} \longrightarrow \mathcal{N}$ . . . . . *Cite as a numeral*
`asIDE`: $\mathcal{V} \longrightarrow \mathcal{I}$ . . . . . *Cite as an identifier*
`asLST`: $\mathcal{V} \longrightarrow \mathcal{L}$ . . . . . *Cite as a list*
`asAPL`: $\mathcal{V} \longrightarrow \mathcal{A}$ . . . . . *Cite as an application*
`asFTN`: $\mathcal{V} \longrightarrow \mathcal{F}$ . . . . . *Cite as a function*
`asERR`: $\mathcal{V} \longrightarrow \mathcal{M}$ . . . . . *Error*

Access to Composite Objects

`_hd`: $\mathcal{V} \longrightarrow \mathcal{V}'$ . . . . . *Head of a composite object*
`_tl`: $\mathcal{V} \longrightarrow \mathcal{V}'$ . . . . . *Tail of a composite object*

Character Manipulation

`Chr?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Character test*
`ChrAsNml`: $\mathcal{C} \longrightarrow \mathcal{N}$ . . . . . *Character's numeric code*
`NmlAsChr`: $\mathcal{N} \longrightarrow \mathcal{C}$ . . . . . *Convert a numeral to a character*

Character Classification

`ScnSPC?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Space-character test*
`ScnDGT?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Digit-character test*
`ScnLFA?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Alpha-character test*
`ScnNON?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Neutral-character test*
`ScnSYM?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Symbol-character test*
`ScnCTL?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Control-character test*

Sequencing

`crc_hd`: $\mathcal{V} \longrightarrow \mathcal{V}$ . . . . . *Coerce-head*
`crc_tl`: $\mathcal{V} \longrightarrow \mathcal{V}$ . . . . . *Coerce-tail*
`seq`: $[\,\mathcal{V}_0\ \mathcal{V}_1\ \cdots\ \mathcal{V}_n\,] \longrightarrow \mathcal{V}_n$ . . . . . *Sequencer*

Interface Operations

`console`: $\mathcal{I} \longrightarrow [\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,]$ . . . . . *Interactive input*
`screen`: $[\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,] \longrightarrow []$ . . . . . *Interactive output*
`dski`: $\mathcal{I} \longrightarrow [\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,]$ . . . . . *File input*
`dsko`: $[\,\mathcal{I}[\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,]\,] \longrightarrow []$ . . . . . *File output*

Text Generation

`issue`: $\mathcal{V} \longrightarrow [\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,]$ . . . . . *Generate text*

Scanning

`scan`: $[\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,] \longrightarrow [\,\mathcal{A}\ \mathcal{C}_j\ \mathcal{C}_{j+1}\ \cdots\,]$ . . . . . *Scan text*
`scans`: $[\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,] \longrightarrow [\,\mathcal{A}_0\ \mathcal{A}_1\ \cdots\,]$ . . . . . `scan` *iterated*

Parsing

`parse`: $[\,\mathcal{C}_0\ \mathcal{C}_1\ \cdots\,] \longrightarrow [\,\mathcal{E}_0\ \mathcal{E}_1\ \cdots\,]$ . . . . `xparses` $\circ$ `scans`
`xparse`: $[\,\mathcal{T}_0\ \mathcal{T}_1\ \cdots\,] \longrightarrow [\,\mathcal{E}\ \mathcal{T}_j\ \mathcal{T}_{j+1}\ \cdots\,]$ . . . . *Parse text*
`xparses`: $[\,\mathcal{T}_0\ \mathcal{T}_1\ \cdots\,] \longrightarrow [\,\mathcal{E}_0\ \mathcal{E}_1\ \cdots\,]$ . . . . `parse` *iterated*

Special Operations

`val`: $\mathcal{E} \longrightarrow \mathcal{V}$ . . . . . *Evaluate*
`evlst`: $[\,\mathcal{E}_0\ \mathcal{E}_1\ \cdots\,] \longrightarrow [\,\mathcal{V}_0\ \mathcal{V}_1\ \cdots\,]$ . . . . . `val` *iterated.*
`let`: $[\,\mathcal{X}\ \mathcal{E}_1\ \mathcal{E}_2\,] \longrightarrow \mathcal{V}$ . . . . . *Lexical binder*
`rec`: $[\,\mathcal{X}\ \mathcal{E}_1\ \mathcal{E}_2\,] \longrightarrow \mathcal{V}$ . . . . . *Recursive binder*
`fix`: $[\,\mathcal{X}\ \mathcal{E}\,] \longrightarrow \mathcal{V}$ . . . . . *Recursive binder*

Daisy Operations
(by group)

Types in descriptors		
	$\mathcal{B}$ – `T` *or* `[]`	
$\mathcal{N}$ – a numeral	$\mathcal{L}$ – a list	$\mathcal{C}$ – a character
$\mathcal{I}$ – a literal	$\mathcal{A}$ – an atom	$\mathcal{E}$ – an expression
	$\mathcal{V}$ – a value	

`add`: `[` $\mathcal{N}_1$ $\mathcal{N}_2$ `]` $\longrightarrow \mathcal{N}_3$ . . . . . *Add*
`all?`: `[` $\mathcal{V}_0$ $\mathcal{V}_1$ $\cdots$ `]` $\longrightarrow \mathcal{B}$ . . . . . *Test for no null elements*
`and`: `[` $\mathcal{N}_1$ $\mathcal{N}_2$ `]` $\longrightarrow \mathcal{N}_3$ . . . . . *Logical-and*
`any?`: `[` $\mathcal{V}_0$ $\mathcal{V}_1$ $\cdots$ `]` $\longrightarrow$ `[` $\mathcal{V}_i$ $\mathcal{V}_{i+1}$ $\cdots$ `]` *Locate a non-Nil element*
`asAPL`: $\mathcal{V} \longrightarrow \mathcal{A}$ . . . . . *Cite as an application*
`asDCT`: $\mathcal{V} \longrightarrow \mathcal{D}$ . . . . . *Cite as a directive*
`asERR`: $\mathcal{V} \longrightarrow \mathcal{M}$ . . . . . *Error*
`asFTN`: $\mathcal{V} \longrightarrow \mathcal{F}$ . . . . . *Cite as a function*
`asIDE`: $\mathcal{V} \longrightarrow \mathcal{I}$ . . . . . *Cite as an identifier*
`asLST`: $\mathcal{V} \longrightarrow \mathcal{L}$ . . . . . *Cite as a list*
`asNML`: $\mathcal{V} \longrightarrow \mathcal{N}$ . . . . . *Cite as a numeral*
`Chr?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Character test*
`ChrAsNml`: $\mathcal{C} \longrightarrow \mathcal{N}$ . . . . . *Character's numeric code*
`console`: $\mathcal{I} \longrightarrow$ `[` $\mathcal{C}_0$ $\mathcal{C}_1$ $\cdots$ `]` . . . . . *Interactive input*
`cons`: `[` $\mathcal{V}_1$ $\mathcal{V}_2$ `]` $\longrightarrow$ `[` $\mathcal{V}_1$ `!` $\mathcal{V}_2$ `]` . . . . . *List constructor*
`crc_hd`: $\mathcal{V} \longrightarrow \mathcal{V}$ . . . . . *Coerce-head*
`crc_tl`: $\mathcal{V} \longrightarrow \mathcal{V}$ . . . . . *Coerce-tail*
`dcr`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Decrement*
`div`: `[` $\mathcal{N}_1$ $\mathcal{N}_2$ `]` $\longrightarrow \mathcal{N}_3$ . . . . . *Divide*
`dski`: $\mathcal{I} \longrightarrow$ `[` $\mathcal{C}_0$ $\mathcal{C}_1$ $\cdots$ `]` . . . . . *File input*
`dsko`: `[` $\mathcal{I}$ `[` $\mathcal{C}_0$ $\mathcal{C}_1$ $\cdots$ `]` `]` $\longrightarrow$ `[]` . . . . . *File output*

`eq?`: $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{B}$ . . . . . *Numeric equality*

`evlst`: $[\mathcal{E}_0\ \mathcal{E}_1\ \cdots] \longrightarrow [\mathcal{V}_0\ \mathcal{V}_1\ \cdots]$ . . . . . `val` *iterated.*

`fix`: $[\mathcal{X}\ \mathcal{E}] \longrightarrow \mathcal{V}$ . . . . . *Recursive binder*

`frons`: $[\mathcal{V}\ \mathcal{V}'] \longrightarrow \mathcal{L}$ . . . . . *Multiset constructor*

`ge?`: $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{B}$ . . . . . *At-least*

`gt?`: $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{B}$ . . . . . *Greater-than*

`head`: $[\mathcal{V}_1\ !\ \mathcal{V}_2] \longrightarrow \mathcal{V}_1$ . . . . . *Head of a list*

`_hd`: $\mathcal{V} \longrightarrow \mathcal{V}'$ . . . . . *Head of a composite object*

`if`: $[\mathcal{T}_0\ \mathcal{V}_0\ \mathcal{T}_1\ \mathcal{V}_1\ \cdots] \longrightarrow \mathcal{V}_i$ . . . . . *Conditional operation*

`in?`: $[\mathcal{U}\ [\mathcal{V}_0\ \mathcal{V}_1\ \cdots]] \longrightarrow \mathcal{B}$ . . . . . *Membership Test*

`inc`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Increment*

`inv`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Invert*

`isAPL?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a application-object*

`isAtm?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for an Atom*

`isDCT?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a directive*

`isERR?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for an identifier-object*

`isFTN?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a function-object*

`isIDE?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a identifier-object*

`isLST?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a list-object*

`isLST?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a list*

`isLtrl?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a literal*

`isNML?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a numeral*

`isNML?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for a numeral*

`issue`: $\mathcal{V} \longrightarrow [\mathcal{C}_0\ \mathcal{C}_1\ \cdots]$ . . . . . *Generate text*

`le?`: $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{B}$ . . . . . *At-most*

`let`: $[\mathcal{X}\ \mathcal{E}_1\ \mathcal{E}_2] \longrightarrow \mathcal{V}$ . . . . . *Lexical binder*

`lt?`: $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{B}$ . . . . . *Less-than*

`mpy`: $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{N}_3$ . . . . . *Multiply*

`ne?`: $[\mathcal{N}_1\ \mathcal{N}_2] \longrightarrow \mathcal{B}$ . . . . . *Numeric inequality*

`neg?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for negative*

`neg`: $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ . . . . . *Negate*

`nil?`: $\mathcal{V} \longrightarrow \mathcal{B}$ . . . . . *Test for Nil*

`NmlAsChr`: $\mathcal{N} \longrightarrow \mathcal{C}$ . . . . . *Convert a numeral to a character*

`one?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for One*
`or`: [$\mathcal{N}_1$ $\mathcal{N}_2$] $\longrightarrow \mathcal{N}_3$ . . . . . *Logical-or*
`parse`: [$\mathcal{C}_0$ $\mathcal{C}_1$ $\cdots$] $\longrightarrow$ [$\mathcal{E}_0$ $\mathcal{E}_1$ $\cdots$] . . . . `xparses` $\circ$ `scans`
`pos?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for positive*
`rec`: [$\mathcal{X}$ $\mathcal{E}_1$ $\mathcal{E}_2$] $\longrightarrow \mathcal{V}$ . . . . . *Recursive binder*
`rem`: [$\mathcal{N}_1$ $\mathcal{N}_2$] $\longrightarrow \mathcal{N}_3$ . . . . . *Remainder*
`same?`: [$\mathcal{U}$ $\mathcal{V}_1$ $\cdots$ $\mathcal{V}_n$] $\longrightarrow \mathcal{B}$ . . . . . *Reference equality*
`scans`: [$\mathcal{C}_0$ $\mathcal{C}_1$ $\cdots$] $\longrightarrow$ [$\mathcal{A}_0$ $\mathcal{A}_1$ $\cdots$] . . . . . `scan` *iterated*
`scan`: [$\mathcal{C}_0$ $\mathcal{C}_1$ $\cdots$] $\longrightarrow$ [$\mathcal{A}$ $\mathcal{C}_j$ $\mathcal{C}_{j+1}$ $\cdots$] . . . . . *Scan text*
`ScnCTL?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Control-character test*
`ScnDGT?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Digit-character test*
`ScnLFA?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Alpha-character test*
`ScnNON?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Neutral-character test*
`ScnSPC?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Space-character test*
`ScnSYM?`: $\mathcal{C} \longrightarrow \mathcal{B}$ . . . . . *Symbol-character test*
`screen`: [$\mathcal{C}_0$ $\mathcal{C}_1$ $\cdots$] $\longrightarrow$ [] . . . . . *Interactive output*
`seq`: [$\mathcal{V}_0$ $\mathcal{V}_1$ $\cdots$ $\mathcal{V}_n$] $\longrightarrow \mathcal{V}_n$ . . . . . *Sequencer*
`sgn`: $\mathcal{N} \longrightarrow$ `-1` or `1` . . . . . *Sign projection*
`sub`: [$\mathcal{N}_1$ $\mathcal{N}_2$] $\longrightarrow \mathcal{N}_3$ . . . . . *Subtract*
`TagOf`: $\mathcal{V} \longrightarrow \mathcal{N}$ . . . . . *Numeric value of a tag*
`tail`: [$\mathcal{V}_1$ `!` $\mathcal{V}_2$] $\longrightarrow \mathcal{V}_1$ . . . . . *Tail of a list*
`_tl`: $\mathcal{V} \longrightarrow \mathcal{V}'$ . . . . . *Tail of a composite object*
`val`: $\mathcal{E} \longrightarrow \mathcal{V}$ . . . . . *Evaluate*
`xor`: [$\mathcal{N}_1$ $\mathcal{N}_2$] $\longrightarrow \mathcal{N}_3$ . . . . . *Exclusive-or*
`xparses`: [$\mathcal{T}_0$ $\mathcal{T}_1$ $\cdots$] $\longrightarrow$ [$\mathcal{E}_0$ $\mathcal{E}_1$ $\cdots$] . . . . `parse` *iterated*
`xparse`: [$\mathcal{T}_0$ $\mathcal{T}_1$ $\cdots$] $\longrightarrow$ [$\mathcal{E}$ $\mathcal{T}_j$ $\mathcal{T}_{j+1}$ $\cdots$] . . . . *Parse text*
`zero?`: $\mathcal{N} \longrightarrow \mathcal{B}$ . . . . . *Test for Zero*

arithmetic operations **Arithmetic Operations**

`add:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	add	$\mathcal{N}_3 = \mathcal{N}_1 + \mathcal{N}_2$
`sub:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	subtract	$\mathcal{N}_3 = \mathcal{N}_1 - \mathcal{N}_2$
`div:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	divide	$\mathcal{N}_3 = \mathcal{N}_1 \div \mathcal{N}_2$
`mpy:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	multiply	$\mathcal{N}_3 = \mathcal{N}_1 \cdot \mathcal{N}_2$
`rem:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	remainder	$\mathcal{N}_3 = \mathcal{N}_1 - (\mathcal{N}_1 \div \mathcal{N}_2) \cdot \mathcal{N}_2$
`inc:`$\mathcal{N}\ \rightarrow\ \mathcal{N}'$	increment	$\mathcal{N}' = \mathcal{N} + 1$
`dcr:`$\mathcal{N}\ \rightarrow\ \mathcal{N}'$	decrement	$\mathcal{N}' = \mathcal{N} - 1$
`neg:`$\mathcal{N}\ \rightarrow\ \mathcal{N}'$	negate	$\mathcal{N}' = -\mathcal{N}$
`inv:`$\mathcal{N}\ \rightarrow\ \mathcal{N}'$	invert	$\mathcal{N}' = \overline{\mathcal{N}}$
`sgn:`$\mathcal{N}\ \rightarrow\ \mathcal{N}'$	sign	$\mathcal{N}' = \begin{cases} \texttt{1}, & \text{if } \mathcal{N} \geq 0 \\ \texttt{-1}, & \text{if } \mathcal{N} < 0 \end{cases}$
`and:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	binary-and	$\mathcal{N}_3 = \mathcal{N}_1 \odot \mathcal{N}_2$
`or:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	binary-or	$\mathcal{N}_3 = \mathcal{N}_1 \oplus \mathcal{N}_2$
`xor:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{N}_3$	exclusive-or	$\mathcal{N}_3 = \mathcal{N}_1 \otimes \mathcal{N}_2$
`zero?:`$\mathcal{N}\ \rightarrow\ \mathcal{B}$	is-zero?	$\mathcal{N} = \texttt{0}$?
`one?:`$\mathcal{N}\ \rightarrow\ \mathcal{B}$	is-one?	$\mathcal{N} = \texttt{1}$?
`neg?:`$\mathcal{N}\ \rightarrow\ \mathcal{B}$	negative?	$\mathcal{N} < \texttt{0}$?
`pos?:`$\mathcal{N}\ \rightarrow\ \mathcal{B}$	positive?	$\mathcal{N} \geq \texttt{0}$?
`lt?:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{B}$	less-than	$\mathcal{N}_1 < \mathcal{N}_2$?
`le?:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{B}$	at-most	$\mathcal{N}_1 \leq \mathcal{N}_2$?
`eq?:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{B}$	equal	$\mathcal{N}_1 = \mathcal{N}_2$?
`ne?:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{B}$	unequal	$\mathcal{N}_1 \neq \mathcal{N}_2$?
`ge?:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{B}$	at-most	$\mathcal{N}_1 \geq \mathcal{N}_2$?
`gt?:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{B}$	greater-than	$\mathcal{N}_1 > \mathcal{N}_2$?
`lt?:`$[\mathcal{N}_1\ \mathcal{N}_2]\ \rightarrow\ \mathcal{B}$	less-than	$\mathcal{N}_1 < \mathcal{N}_2$?

Errons

|—`any/`—| *Erroneous list (* `all?` *).*
|—`any/`—| *Erroneous list (* `any?` *).*
|—`arg/`—| *Invalid formal argument.*
|—`chr/`—| *Non-character operand.*
|—`cmp/`—| *Erroneous comparison (* `same?` *or* `in?` *).*
|—`crc/`—| *Invalid coercion.*
|—`dfn/`—| *Invalid assignment.*
|—`dvc/`—| *Device error.*
|—`evl/`—| *Non-list argument (* `evlst` *).*
|—`f-c/`—| *Construction error*
|—`ftn/`—| *Erron applied.*
|—`hd?/`—| *Invalid head-access*
|—`ifA/`—| *Invalid alternative (* `if` *).*
|—`ifP/`—| *Erroneous predication (* `if` *).*
|—`ld?/`—| *Invalid access*
|—`lst/`—| *Erron in list-expression.*
|—`opn/`—| *Invalid operation code.*
|—`nla/`—| *Non-list operand (* `head` *or* `tail` *).*
|—`nn0/`—| *Non-numeric operand.*
|—`nn1/`—| *Non-numeric operand.*
|—`prb/`—| *Invalid numeric probe.*
|—`sam/`—| *Invalid argument (* `same?` *or* `in?` *).*
|—`scn/`—| *Invalid argument (* `scan`, `scans` *).*
|—`seq/`—| *Invalid argument (* `seq` *).*
|—`scn/`—| *Invalid argument (* `parse`, `xparse`, `xparses` *).*
|—`sc0/`—| *Invalid text (* `scan`, `scans` *).*
|—`sc1/`—| *Erroneous text (* `scan`, `scans` *).*
|—`tag/`—| *Invalid citation.*
|—`tl?/`—| *Invalid tail-access*
|—`trj/`—| *Invalid trajectory.*
|—`ubi:`—| *Unbound identifier.*
|—`val/`—| *Value-of-erron.*
|—`xps/`—| *Transposition error.*

Syntax Errons

Syntax Erron Indicators

Indicator	Object
`d`	a directive
`n`	a numeral
`m`	an erron
`i`	an identifer object
`l`	a list object
`a`	an application object
`f`	a function object

Parsing Messages

star is one of the indicators above.

Message	Explanation
`.`—	Message fill
`..`—	Message fill
`...`—	Message fill
`^`—	Error in value quotation
`(`—	Error in parenthesization
⋆`:`—	Error in application's argument
`\`—	Error in formal argument
`\`⋆`.`—	Error in function body
`[`—	Error before matching '`···]`'
`<`—	Error before matching '`···>`'
`{`—	Error before matching '`···}`'
`!`—	Error after a '`!`'
`@'='`	Error (nonliteral) before '`=`'
⋆`=`—	Error after an '`=`'

—NOTES—