

Pointfrip Quickinfo

2023-09-08

The following is about programming at the **function-level** with combinators

Rule

As a rule, **right-before-left** applies, but there are exceptions, e.g. in condition terms.

Parentheses must be used to change the evaluation of the terms.

Infix notation applies as in: $a + b$

For functions you write: $function \circ argument$

Data Types

$[0], [1], [2], \dots, [i],$
 $[_{123}]$ are selectors that access the values of a list
or a dict -- or are integer* numbers

name is an identifier for an associated function

$_{123.5678e_{30}}$ is a real number

$(10 ; 20 ; 30 ; 40 ; 50 ;)$ is a list of real numbers

$(10\ a\ 20\ b\ 30\ c\ 40\ d\ 50\ e)$ is a dict* with values and keys

$()$ empty list / *null*

$(head\ infix\ ..\ tail)$ data cell / *prop*

"abcdef" is a string

true / false are of type *bool*

*note that the constant combinator should be used.

Definition of Functions/Constants/Operators

<i>identifier == term</i>	assigns a term to the identifier
<i>constname == ' literal</i>	Constants use the constant combinator
<i>oprname == (...) ° ee</i>	Operators often use an ee and [0] and [1]

Combinators

<i>'name</i>	is the Constant combinator
<i>function1 ° function2</i>	is the Composition, o can also be used (right-pipe)
<i>fun1 , fun2 , ... , funm ,</i>	is the Construction of a list
<i>(test -> then ; else)</i>	is the Condition combinator with an alternative
<i>(test ->* term)</i>	is a While loop
<i>(function aa)</i>	is the Apply-to-All combinator (map)
<i>(function \)</i>	is the Inserttr combinator (reduce)
<i>function1 ee function2</i>	evaluates the functions and creates a pair from them
<i>#name</i>	picks the value for the name from a dict
<i>function : argument</i>	is an Application -- <i>function(argument)</i>
<i>liste insl 'func</i>	is the Insertl operator; insr for Inserttr operation
<i>func _s</i>	Single function is executed
<i>func1 app func2</i>	Apply function to execute Functionals
<i>func1 swee func2</i>	like ee , only the elements in the pair are swapped
<i>(func aa0) ° liste,x,y, ... ,</i>	Mixture of aa and distr , expanded
<i>(liste,x,y, ... ,) map0 'func</i>	Mixture of map and distr , extended
<i>liste filter 'boolfunc</i>	is the Filter operator

Listenverarbeitende Funktionen und Operatoren

<i>val0 ; val1 ; val2 ; ... ;</i>	Listenbildung mit literalen Werten
head ° <i>prop</i>	extrahiert den ersten Wert einer Liste
tail ° <i>prop</i>	extrahiert den Rest einer Liste
infix ° <i>prop</i>	extrahiert den Infixwert einer Liste/Dicts
prop ° <i>hd, inf, tl,</i>	erzeugt eine Datenzeile mit drei Werten
term ° <i>combi</i>	extrahiert den Term-Wert aus einem Combine-Datentyp
arg ° <i>combi</i>	extrahiert den Arg-Wert aus einem Combine-Datentyp
type ° <i>data</i>	liefert eine Bezeichnung für den Datentyp
<i>liste</i> at <i>num</i>	pickt den <i>num</i> -ten Wert aus der <i>liste</i>
<i>func</i> , <i>list</i>	das Komma fügt ein Element vor die Liste
iota ° <i>num</i>	erzeugt eine Liste von Zahlen ab 1 aufwärts bis <i>num</i>
<i>num1</i> to <i>num2</i>	erzeugt eine Liste von Zahlen von <i>num1</i> bis <i>num2</i>
reverse ° <i>liste</i>	kehrt eine Liste um; funktioniert auch mit einem String
trans ° <i>matrix</i>	Transpose einer Liste von Listen (matrix)
<i>data</i> distl <i>liste</i>	Distribution Left
<i>liste</i> distr <i>data</i>	Distribution Right
<i>data</i> make <i>num</i>	erzeugt eine Liste mit <i>num</i> <i>data</i> -Werten
<i>liste</i> take <i>num</i>	liefert eine Liste der ersten <i>num</i> Elemente
<i>liste</i> drop <i>num</i>	liefert die Restliste ohne die ersten <i>num</i> Elemente
<i>liste1</i> ++ <i>liste2</i>	liefert eine verkettete Liste
length ° <i>liste</i>	liefert die Länge einer Liste
<i>liste</i> count <i>data</i>	liefert die Anzahl von <i>data</i> in der Liste
<i>liste</i> find <i>data</i>	liefert die erste Position von <i>data</i> in der Liste

Numerische Funktionen und Operatoren

$num1 + num2$	Addition von Zahlen gleichen Typs
$num1 - num2$	Subtraktion von Zahlen gleichen Typs
$num1 * num2$	Multiplikation von Zahlen gleichen Typs
$num1 / num2$	Division von Zahlen gleichen Typs
$num1 \wedge num2$	Potenzierung von Zahlen gleichen Typs
$num1 \text{ idiv } num2$	Division von Integerzahlen
$num1 \text{ imod } num2$	Modulo von Integerzahlen
$\text{pred} \circ num$	Vorgängerfunktion
$\text{succ} \circ num$	Nachfolgerfunktion
$\text{sign} \circ num$	Signumfunktion
$\text{abs} \circ num$	Betrag der Zahl
$\text{neg} \circ num$	Negation der Zahl
$_ \circ num$	Negation der Zahl
$\text{floor} \circ num$	Abrunden der Zahl
$\text{ceil} \circ num$	Aufrunden der Zahl
$\text{float} \circ num$	wandelt in eine Realzahl um
$\text{round} \circ num$	rundet in eine Integerzahl um
$\text{trunc} \circ num$	Integerzahl ohne Beachtung der Nachkommawerte
$\text{real roundto } num$	Rundet auf die num -te Nachkommastelle

exp ° num	Exponentialfunktion der Zahl
ln ° num	natürlicher Logarithmus der Zahl
lg ° num	Zehnerlogarithmus der Zahl
sq ° num	das Quadrat der Zahl
sqrt ° num	die Quadratwurzel der Zahl
cbrt ° num	die Qubikwurzel der Zahl
pi	Funktion liefert die Zahl Pi
2pi	Funktion liefert den Umfang des Einheitskreises
sin ° num	Sinusfunktion der Zahl in Radiant
cos ° num	Cosinusfunktion der Zahl in Radiant
tan ° num	Tangensfunktion der Zahl in Radiant
arcsin ° num	Arcussinusfunktion
arccos ° num	Arcuscosinusfunktion
arctan ° num	Arcustangensfunktion
y arctan2 x	Phase (oder Arg) zu (x,y)
sinh ° num	Sinus Hyperbolicus Funktion
cosh ° num	Cosinus Hyperbolicus Funktion
tanh ° num	Tangens Hyperbolicus Funktion
deg ° num	wandelt Radiant in Degree um
rad ° num	wandelt Degree in Radiant um

Boolische Funktionen und Operatoren

<i>data1</i> = <i>data2</i>	prüft auf Gleichheit
<i>data1</i> != <i>data2</i>	prüft auf Ungleichheit
<i>data1</i> <> <i>data2</i>	prüft auf Ungleichheit, alternativ
<i>data1</i> < <i>data2</i>	Vergleich auf Kleiner-als
<i>data1</i> > <i>data2</i>	Vergleich auf Größer-als
<i>data1</i> <= <i>data2</i>	Vergleich auf Kleiner-Gleich
<i>data1</i> >= <i>data2</i>	Vergleich auf Größer-Gleich
<i>data1</i> min <i>data2</i>	Minimum von <i>data1</i> und <i>data2</i>
<i>data1</i> max <i>data2</i>	Maximum von <i>data1</i> und <i>data2</i>
not ° <i>bool</i>	Boolische Nicht-Funktion
<i>bool1</i> and <i>bool2</i>	Boolische Und-Funktion
<i>bool1</i> or <i>bool2</i>	Boolische Oder-Funktion
<i>bool1</i> xor <i>bool2</i>	Boolische Exklusiv-Oder-Funktion
isatom ° <i>data</i>	Prüft, ob <i>data</i> zu den Atom-Typen gehört
isnull ° <i>data</i>	Prüft, ob <i>data</i> der Wert () also Null ist
isprop ° <i>data</i>	Prüft, ob <i>data</i> eine Datenzelle ist
islist ° <i>data</i>	Prüft, ob <i>data</i> eine Liste ist
isnum ° <i>data</i>	Prüft, ob <i>data</i> eine Zahl ist, generisch
iszero ° <i>data</i>	Prüft, ob <i>data</i> die Zahl 0 ist, generisch
ispos ° <i>num</i>	Prüft, ob <i>num</i> eine positive Zahl ist, generisch
isneg ° <i>num</i>	Prüft, ob <i>num</i> eine negative Zahl ist, generisch
isident ° <i>data</i>	Prüft, ob <i>data</i> ein Bezeichner ist

isint ° <i>data</i>	Prüft, ob <i>data</i> eine Integerzahl ist
isreal ° <i>data</i>	Prüft, ob <i>data</i> eine Realzahl ist
isstring ° <i>data</i>	Prüft, ob <i>data</i> ein Zeichenstring ist
iscons ° <i>data</i>	Prüft, ob <i>data</i> eine List-Datenzelle ist
isquote ° <i>data</i>	Prüft, ob <i>data</i> ein Quotewert ist
isivar ° <i>data</i>	Prüft, ob <i>data</i> ein Instanz-Variablen-Selektor ist
iscombi ° <i>data</i>	Prüft, ob <i>data</i> ein Combine-Wert ist
isact ° <i>data</i>	Prüft, ob <i>data</i> ein Act-Wert ist
isbool ° <i>data</i>	Prüft, ob <i>data</i> ein boolischer Wert ist
isbound ° <i>ident</i>	Prüft, ob der Bezeichner schon definiert ist
isundef ° <i>data</i>	Prüft, ob <i>data</i> der Wert _undef ist
<i>data</i> in <i>liste</i>	Prüft, ob <i>data</i> als Element in der <i>liste</i> enthalten ist

Dict Funktionen und Operatoren

#ident ° <i>dict</i>	der Selektor pickt zum <i>ident</i> -Key den Wert aus dem <i>dict</i>
<i>dict</i> iget <i>ident</i>	zum <i>ident</i> *-Key wird der Wert aus dem <i>dict</i> herausgepickt
<i>dict</i> iput <i>ident</i> , <i>value</i> ,	zum <i>ident</i> *-Key wird der <i>value</i> neu im <i>dict</i> angelegt
<i>dict</i> get <i>key</i>	zum <i>key</i> wird der Wert aus dem <i>dict</i> herausgepickt
<i>dict</i> put <i>key</i> , <i>value</i> ,	zum <i>key</i> wird der <i>value</i> neu im <i>dict</i> angelegt
(<i>ident</i> := <i>func</i>) ° <i>dict</i>	wie bei iput geschieht diese "Variablen"-Zuweisung
(<i>func</i> <- <i>x</i> ; <i>y</i> ; ... ;) ° <i>liste</i>	<i>func</i> wendet das erzeugte Dict, wie nach einem Assign, an
keys ° <i>dict</i>	erzeugt eine Liste mit allen Keys aus dem <i>dict</i>
values ° <i>dict</i>	erzeugt eine Liste mit allen Values aus dem <i>dict</i>
it ° <i>dict</i>	Pickt aus dem <i>dict</i> den zu _it zugehörigen Wert

String Funktionen und Operatoren

length ° <i>string</i>	gibt die Länge des Strings an
substring ° <i>string, i, len,</i>	Kopiert einen Teilstring aus <i>string</i>
<i>string1</i> & <i>string2</i>	verkettet zwei Strings
<i>string1</i> concat <i>string2</i>	verkettet zwei Strings
<i>string</i> indexof <i>substr</i>	sucht die Position von <i>substr</i> im <i>string</i> von links
trim ° <i>string</i>	schneidet links und rechts die Leerzeichen ab
triml ° <i>string</i>	schneidet links die Leerzeichen ab
trimr ° <i>string</i>	schneidet rechts die Leerzeichen ab
upper ° <i>string</i>	wandelt den String in Großbuchstaben um
lower ° <i>string</i>	wandelt den String in Kleinbuchstaben um
capitalize ° <i>string</i>	wandelt den String in ein Hauptwort um
char ° <i>num</i>	erzeugt ein Zeichen nach dem Unicodewert
unicode ° <i>string</i>	bestimmt den Unicodewert des ersten Zeichens
parse ° <i>string</i>	parst den String mit dem Pointfrip-Parser
value ° <i>string</i>	wandelt Zahlen, Wörter,Listen im String in Daten um
string ° <i>data</i>	wandelt die Daten in einen Printstring um
unpack ° <i>string</i>	zerlegt den String in eine Liste von Einzelzeichen
<i>string</i> split <i>delstr</i>	zerlegt den <i>string</i> in eine Liste mit Strings ohne <i>delstr</i>
<i>liste</i> join <i>insstr</i>	verbindet die Strings der Liste mit <i>insstr</i> dazwischen

Matrix Funktionen und Operatoren

<i>matrix1</i> add <i>matrix2</i>	Addiert zwei Matrizen, komponentenweise
<i>matrix1</i> sub <i>matrix2</i>	Subtrahiert <i>matrix2</i> von <i>matrix1</i>
<i>matrix1</i> mul <i>matrix2</i>	Multipliziert zwei Matrizen
<i>num</i> mul <i>matrix</i> <i>matrix</i> mul <i>num</i>	Multipliziert die <i>matrix</i> mit einem Skalarwert
ismat ° <i>data</i>	Prüft, ob <i>data</i> eine Matrix ist, vereinfachte Form
trans ° <i>matrix</i>	Transpose der <i>matrix</i>
det ° <i>matrix</i>	Berechnet die Determinante der <i>matrix</i>
inv ° <i>matrix</i>	Berechnet die Inverse Matrix
<i>num1</i> zeromat <i>num2</i>	erzeugt eine Matrix mit lauter Nullen
idmat ° <i>num</i>	Identitätsmatrix der Größe <i>num</i>
fail ° <i>infodata</i>	erzeugt Standardfehlermeldung für einen Fail
<i>liste</i> IP <i>liste</i>	Inner Product nach John Backus
<i>matrix</i> MM <i>matrix</i>	Matrixmultiplikation nach John Backus
rnd ° <i>matrix</i>	Rundet <i>matrix</i> auf fünf Nachkommastellen
zero ° <i>data</i> zero ° <i>matrix</i>	generiert eine Null, typenabhängig
one ° <i>data</i> one ° <i>matrix</i>	generiert eine Eins, typenabhängig

Misc Funktionen und Operatoren

undef	erzeugt Fehlermeldung für undefinierte Funktion
id ° <i>data</i>	Identitätsfunktion liefert <i>data</i>
name ° <i>ident</i>	extrahiert den String des Bezeichners
body ° <i>ident</i>	extrahiert den Definitionswert des Bezeichners
info ° <i>ident</i>	extrahiert den Compilerstring des Bezeichners
identlist	gibt eine Liste mit allen verwendeten Bezeichnern aus
quote ° <i>data</i>	macht aus <i>data</i> einen Quote-Wert
<i>ident</i> error <i>string</i>	gibt eine Fehlermeldung mit <i>ident</i> und <i>string</i> aus
<i>'func1</i> comp <i>'func2</i>	verkettet die Funktionen zu einer neuen Funktion
<i>int</i> act <i>dict</i>	erzeugt einen Act-Wert mit den Daten - (Monade)
<i>act</i> bind <i>'func</i>	legt die <i>func</i> im bind-Feld von einem neuen <i>act</i> an
<i>act</i> >> <i>func</i>	legt die <i>func</i> im bind-Feld von <i>einem neuen act</i> an
<i>fname</i> load	liest den Text von der Datei <i>fname</i> ins Display
<i>fname</i> save	sichert den Text vom Display in die Datei <i>fname</i>
files	gibt eine Liste mit allen Dateinamen aus
<i>fname</i> loadtext	lädt den String aus der Datei <i>fname</i> im Ordner "pf/"
<i>fname</i> savetext <i>string</i>	sichert den <i>string</i> in der Datei <i>fname</i> im Ordner "pf/"
stopvm	bricht die Berechnung ab, mit Fehlermeldung
dump	zeigt alle Bezeichner mit ihren Zuordnungen an
savedump (for test)	zeigt alle info-Strings der Bezeichner an
help	Linkadresse zu aktuellem Hilfe-PDF
pim ° <i>num</i>	gibt eine Liste aller Primfaktoren einer Zahl an, Beispiel
(<i>test</i> try then;else)° <i>argum</i>	Prüft <i>test</i> auf Error -> then/else mit (<i>result</i> ; <i>argum</i> ;)

Hinweise zum Laden und Speichern von Programmdateien

"filename" save	ein Programmtext wird unter dem Namen <i>filename</i> im Ordner „pf/“ gesichert
"filename" load	ein Programmtext der Datei <i>filename</i> aus dem Ordner „pf/“ wird mit den Definitionen eingelesen
files	gibt Liste mit allen Dateinamen im Ordner „pf/“ aus

Mit **identlist** oder **dump** bekommt man einen Überblick über die verwendeten Wörter.

*man beachte, daß der Konstanten-Kombinator verwendet werden sollte.

(CC0)