Pointfrip Quickinfo

2022-11-29

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* **°** *argument*

Data Types

[0] , [1] , [2] , … , **[** ***i*** **]** , [_123]	are selectors that access the values of a list, dict, or array; or are integers*
name	is an identifier for a function associated with it
_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

*note that the constant combinator should be used.

Definition of Functions/Constants/Operators

identifier **==** *term*	assigns a term to the identifier
constname **==** **'** *literal*	Constants use the constant combinator
operatorname **==** (…) **° ee**	Operators often use an ee and [0] and [1]

Combinators

'*name*	is the constant combinator
function1 **°** *function2*	is the composition
fun1 **,** *fun2* **,** ... **,** *funm* **,**	is the construction of a list
(*test* **->** *then* **;** *else*)	is the condition combinator with an alternative
(*test* **->*** *term*)	is a while loop
(*function* **aa**)	is the apply-to-all combinator (map)
(*function* ****)	is the insert combinator (reduce)
function1 **ee** *function2*	evaluates the functions and creates a pair from them
#*name*	picks the value for the name from a dict

Functions

id	identity function
iota	produces a list of numbers from 1 up to the number
head	extracts the first value of a list
tail	extracts the rest of a list
infix	extracts the infix value of a list/dict
prop	creates a cell of First,Infix,Rest,
top	like head, but not for objects
pop	like tail, but not for objects
tag	extracts the type or infix value
reverse	reverses a list

length	returns the length of a list
sin	calculates the sine of a number
ln	calculates the natural logarithm of a number
islist	checks whether it is a list

Operators

first **,** *rest*	Comma creates a list
num1 **+** *num2*	Arithmetic operators for addition,
num1 **-** *num2*	for subtraction
num1 ***** *num2*	for multiplication
num1 **/** *num2*	for division
dict **get** *key*	determines the value for the key*
dict **put** *key*,*value*,	replaces/creates a key* with value in the dict
(*key* **:=** *value*) ° *dict*	like put, but as a value assignment in the dict
num1 **=** *num2*	checks for equality and then returns **true** or **false**
etc	

Objects and Classes

(turtle **::** ( ) stack 0 x 0 y ...)	is the turtle object with the attributes stack, x, y, ...
turtle == **..** **{** dict ... ... **}**	is the turtle class with the ... methods

Monads and Effects

('turtle new) (**draw** eff 'io) creates a monad for drawing the turtle track

io == .. { … … … } are the system effects, so to speak the driver (?)

et cetera can be found in the reference/blue question mark

*note that the constant combinator should be used.