Principles of Programming Languages - Homework 3

Abhi Agarwal

1 Problem 1

(a)

$\cdot : N \times N \to N$
$\emptyset \cdot y = y$
$y \cdot \{x\} = y + \{y \cdot x\}$

(b)

$\forall x \in N : \{x\} = x + \{\emptyset\}$
1. Prove that $\emptyset$ satisfies A.
x = $\{\emptyset\}$
$\{\{\emptyset\}\}$ = $\{\emptyset\}$ + $\{\emptyset\}$
2. Prove that for all $x \in N$ that satisfy A, {x} satisfies A.
x = $\{x\}$
$\{\{x\}\}$ = $\{x\}$ + $\{\emptyset\}$

(c)

$filter : \mathbb{Z} \times List \to List$
$filter(nil) = nil$
$filter(< hd, tl >) = append(hd, filter(n, tl))\ if\ hd >= n$
$filter(< hd, tl >) = filter(n, tl)$ otherwise

(d)

Provided in *.scala* file.

(e)

2 Problem 2

(a)

string values to numerical values
1. ‘1’/ 1 (== 1)
2. ‘1’ - ‘3’ (== -2)
3. ‘1’ >>> 2 (== 0)

Boolean values to numerical values
4. true * 2 (== 2)
5. true * null (== 0)
6. true / true (== 1)
7. true & false (== 0)

string values to Boolean values
8. !‘true’ (== false)
9. ‘true’ <= ‘false’ (== false)

numerical values to Boolean values
10. !0 (== true)

Infer general rules that describe how the considered implicit type conversions work

When you’re doing web-development you usually get inputs that are string, but should be either numerical or boolean values. There’s a precedence of everything depending on the operator.

When there’s a minus it does an implicit cast of both of the sides to be a number. Then it tries to give you a numerical value out. If you do ‘1‘ - 1 you would get 0. If you do ‘x‘ - ‘x‘ you would get NaN. This case also applies for division, multiplication, modulus. In this way it’s really easy to do String or Boolean to numerical values using minus, multiplication, division, and modulus.

Then there are certain operators that allow us to go from either String or numerical values to Boolean values. There’s the !, and <=, >=, etc. All of the comparison or value comparison operators. When you have these operators it converts both sides to true or false.

Therefore, depending on the operator or the class of operators we are considering - Node.JS converts both of the sides to being the same type.

(b)

Expressions that involve the + operator

1. ‘x’ + 1 (== ‘x1’) and 1 + ‘x’ (== ‘1x’)
2. +‘3.65’ (== 3.65)
3. 1 + 1 + ‘1’ (== ‘21’)
4. {} + {} (== ‘[object Object][object Object]’)
5. [] + {} (== ‘[object Object]’) and {} + [] (== 0)
6. [] + [] (== ‘’) or [0] + [] (== ‘0’)

Infer a general rule that describes how + is interpreted in Javascript

It seems like you can add only numbers and strings. An object is converted to either a number or a string (depending on which side it is on). Arrays are just converted into strings with all of their elements as strings with commas separating them.

If either of the left or right side of the plus symbol is a string then the other side is automatically casted into a string. If they are both numbers then it adds the numbers together. When two strings are added they are just concatenated together.

If you look at example 5 above, if the left side is an array then it automatically becomes a string. This makes it a concatenation. If the left side is an object then it automatically becomes a number. Given the example {} + ‘x’ becomes NaN, and {} + ‘1’ becomes 1.

(c)

Expressions that involve the && and || operator

1. 1 && true (== true)
2. ‘hello’ && false (== false)
3. true && 1 (== 1) and true || 1 (== true)
4. false || ‘hello’ (== ‘hello’)
5. {} || {} (== {}) and [] || []
6. [] && {} is possible (== {}), but {} && [] is not.

Infer a general rule that describes how the && and || operators are interpreted in Javascript

false && anything is always false. true || anything is always true.

If there are strings or numbers before the || then it is returned as a string or a number. If true is on the left side of || then it returns true. If there is false on the left side of || then it returns anything on the right hand side.

If there are strings or numbers before the && then it returns the variable on the right hand side of the &&. If true is on the left side of the && then it returns anything on the right hand side. If there's false on the left hand side of && then it returns false.