Natural Number Fuzzing

by Sven Nilsen, 2018

The following indeterministic algorithm in Dyon has the property that it can guess a natural number, stored inside a black box that only returns “yes” or “no” whether the number was correct. This function can be studied or used for creating functions that generate sentences of infinite sets, such as languages.

```
nat(d: f64) =
   if (random() < 0.2) || (d <= 0) {1}
   else if random() < 0.5 {nat(d-1) + nat(d-1)}
   else {nat(d-1) * nat(d-1)}
```

It can not guess `0`, so this must be done separately. The argument is a depth value that forces it to terminate when going too deep. The parameter `0.2` is optimized for long run creativity, that is how likely it is to guess a number that has not been tried before. The 30 000 first guesses are likely to contain over 70% new guesses.

The function constructs natural numbers by the following building blocks:

1	`1` is the only constant
+	Addition
*	Multiplication

One can think of natural numbers as a tree with two types of nodes, with `1` in all the leaves. The tree computes a new natural number. Since all branches picks a new natural number using the same algorithm, it means the probability distribution is self-similar.

Since 0.2 probability to terminate a branch is very good for natural numbers, one might try this value when generating sentences of other languages.

The law of distribution of addition over multiplication means that the left side form is more likely to occur than the right side form, since the right side depends on two copies of `a`:

a * (b + c) = a * b + a * c

This knowledge can also be used when generating sentences of other languages: If a sentence is only valid if it contains two or more copies of a generated element, then this element should be factorized out to make the sentence easier to generate.