We can define mathematical operators on numbers as recursive functions. For example addition is the following:

add 0 y = y add s(n) y = s(add n y)

We can write this as the following λ abstraction:

$$\text{add} = \lambda x, y : \text{nat}\,.\; case(x, z \mapsto y, s(v) \mapsto s(\text{add}\;\; v\;\; y))$$

This is a recursive function, so must be the fixpoint of another function A:

$$A = \lambda f : Nat \rightarrow Nat \rightarrow Nat.\; \lambda x, y : \text{nat}\,.\; case(x, z \mapsto y, s(v) \mapsto s(f\;\; v\;\; y))$$

then add $x\ y = (fixA)\ x\ y$

So we write this in PCF as:

$$\text{fix}\, f : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}\,.\; A : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}$$

.

When we try to evaluate this term, we get the following, by the evaluation rule for fix:

$fixf : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}\,.\; A : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat} = [f : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}\,.\; A : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}\,/f]A$. This expands to:

$$\lambda x, y : \text{nat}\,.\; case(x, z \mapsto y, s(v) \mapsto$$
$$s((fixf : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}\,.\; A : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat})\;\; v\;\; y))$$

Therefore, expanding this infinitely gives all possible executions of the addition function.