Existential Paths as Sets

by Sven Nilsen

An existential path is a function that tells what another function returns. It takes an argument of the output type and returns \`true\` if this value is returned and \`false\` otherwise.

For example, a function \`inc\` that increases a natural number returns only numbers larger than zero, since there is no natural number one can increase to get zero:

```
∃inc := \(x : nat) = x ¬= 0

inc := \(x : nat) = x + 1
```

Although this seems intuitively true, I have not checked this for every natural number (since there are infinitely many of them). This is a common problem: For functions that returns very large sets, it is very hard to test whether the existential paths is correct.

Existential paths have properties similar to sets, except that they exists explicitly as code. This makes them more practical than the pure mathematical concept of a set, because they indirectly dictate how hard it is to solve certain problems.

Notice that it is possible to describe many infinite sets, otherwise one could not write down an existential path for a function. However, there are sets that are very complex-looking, such as fractals or composite numbers.

When the existential path takes very long time to compute compared to checking the value by computing some input for the function, it is called a “one-way function”. This is because if there existed an easy way to reverse the function such an input could be found that reproduced the output, then this algorithm could be used by the existential path to compute much faster.

If you can prove that there are existential paths that must take very long time to compute compared to the function, you would prove the famous \`P ¬= NP\` conjecture, become world famous and win a million dollars.