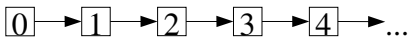
Programming language theory

2 Semantics of natural numbers

2.1 Informal description

The natural numbers $1, 2, 3, \ldots$ can be used even by small children for counting things. In school the number 0 is introduced and it will be convenient to let it be the first natural number. We soon get a picture of the natural numbers like



where 0 is the first number and after each number there is a successor and there is no bound on how far we can extend this picture to the right; there is no largest natural number.

We soon learn to add numbers. Given two disjoint collections we can count the number of elements in each collection. We then form a new collection containing all the things in the given collections. According to our experience no things disappear (and no new things are created) during this process and we say that the number of things in the new collection is the sum of the numbers of things in the two given collections. With this model it is "obvious" that the sum of two numbers is independent of the order in which we give the numbers. Using a formula we "understand" that $n + m = m + n$ for all natural numbers n and m. In this chapter we will formally prove this theorem.

Why should we prove something that we "know" to be true? Well, the important thing is to understand what a theory and a proof really is, and the proof is longer than you might expect.

Some theorems on natural numbers are very hard to prove. In 1993 Andrew Wiles proved that there are no natural numbers $x, y, z > 0$ and $n > 2$ such that $x^n + y^n = z^n$. In the late 16th century the French lawyer and mathematician Pierre de Fermat thought that he had discovered a proof of this fact. The margin of the text book where he made a note of this was too small to contain his proof. Wile's proof is more than 100 pages long. It is believed to be correct even though not many mathematicians have read it.

2.2 Peano's axioms

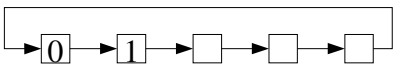
An axiomatic theory for the natural numbers was constructed by Dedekind in 1880 which has become known under the name of Peano's axioms. The axioms introduce three concepts: the *set of natural numbers*, $\mathbb{N}$, the number *zero*, 0, and the *successor* of a natural number denoted by an apostrophe after the number.

1. $0 \in \mathbb{N}$.
2. $n \in \mathbb{N} \Rightarrow n' \in \mathbb{N}$.
3. $n \in \mathbb{N} \Rightarrow n' \neq 0$.
4. If $n, m \in \mathbb{N}$ then $n' = m' \Rightarrow n = m$.

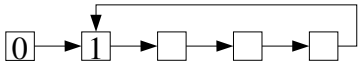
5. If $A \subseteq \mathbb{N}$, $0 \in A$ and $n \in A \Rightarrow n' \in A$ then $A = \mathbb{N}$.

The first two axioms are "constructive" while the last three are "restrictive". The first axiom states that there is a natural number denoted by 0. The second one states that if n is a natural number then n' is a natural number. Hence $0', 0''$ and $0'''$ are natural numbers, usually denoted by 1, 2 and 3.

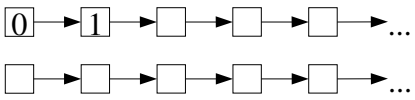
The restrictive axioms describe properties of the natural numbers The third axiom states that 0 is not a successor of any natural number; it is the "first" number. It excludes e.g. the following model:



The fourth axioms states that if two successors are equal they must be the successors to the same number. This rules out



The last axiom states that $\mathbb{N}$ is the smallest set satisfying the other axioms. It is called the *induction axiom* and it is required to prove properties that hold for all natural numbers. Without it the following structure could represent the natural numbers.



Axioms cannot be proved. They are the rules of the game. Using them we can define other concepts and with the rules of logical reasoning we can prove theorems about natural numbers.

We are in a position to define *addition* for natural numbers. We will of course denote the sum of two natural numbers, n and m, by $n + m$.

$0 + m = m$ for all $m \in \mathbb{N}$

$n' + m = (n + m)'$ for all $n, m \in \mathbb{N}$

Before accepting this definition we should prove that it uniquely defines $n + m$ for all n and m. It is a recursive definition and recursive definitions might fail to terminate. It would neither be acceptable if it gives several values for the same sum. A careful proof can be found in E. Landau, *Foundations of Analysis*, Chelsea Publishing Company, 1966.

After accepting it we can prove that $0'' + 0''' = 0''' + 0''$.

$$0'' + 0''' = (0' + 0''')' = (0 + 0''')'' = (0''')'' = 0''''$$
$$0''' + 0'' = (0'' + 0'')' = (0' + 0'')'' = (0 + 0'')''' = (0'')''' = 0''''$$

We will prove one theorem using the axioms.

Sats.[] $n + 0 = n$ for all $n \in \mathbb{N}$. ■

Proof. Let A be the set of all n such that $n + 0 = n$. We are going to use the induction axiom to prove that $A = \mathbb{N}$.

1. $0 \in A$ since $0 + 0 = 0$ by the first clause in the definition of addition.
2. Now assume that $k \in A$. Then $k+0 = 0$ by the definition of A. Further, $k'+0 = (k+0)' = k'$ by the second clause in the definition of addition.

□

We will not prove that $n+m = m+n$ for all n and m directly from the axioms but will prove a corresponding property of a Haskell function for adding natural numbers. It is easy to transform the previous proof to that notation and vice versa.

2.3 Java representation of natural numbers

Even if Java did not contain the `int` data type it would be possible represent and compute with natural numbers as defined in the previous section. It will be terribly inefficient, but very instructive.

```
interface N {
  N add(N m);
  N mul(N m);
}

class Zero implements N {
  N add(N m) {
    return m;
  }
  N mul(N m) {
    // omission
  }
}

class Suc implements N {
  private N n;
  Suc(N n) {
    this.n = n;
  }
  N add(N m) {
    return new Suc(n.add(m));
  }
  N mul(N m) {
    // omission
  }
}
```

We say that this representation is a *model* of $\mathbb{N}$ if the axioms hold for all `N` objects after translating them to Java notation. This will almost be possible.

The three concepts from Peano's axioms reappear.

1. The set $\mathbb{N}$ corresponds to the class `N`. In the current version of Java `null` is also of type `N`. We assume that we are using a Java version without this defect.
2. The number 0 corresponds to a `Zero` object.
3. The successors correspond to `Suc` objects.

Obviously the first two axioms hold; `new Zero()` is of type `N` and `new Suc(n)` is of type `N` if `n` is of type `N`.

For axioms 2 and 3 to be meaningful equality must be defined. The definition `==` and the default implementation of `equals` for Java objects is not adequate since both `new Zero() == new Zero()` and `new Zero().equals(new Zero())` will return `false`. The remedy is to redefine `equals` to make all `Zero` objects equal, and similarly for successor objects. Then both axiom 3 and 4 will hold. We cannot formally prove this without a formal semantics for Java. Also axiom 5 will hold provided that `null` is not of type `N`.

2.4 Haskell representation of natural numbers

Haskell is superior to Java:

```
data N = Zero | Suc N
    deriving Eq
```

The clause `deriving Eq` implements automatically the equality operator the way we want:

```
(Zero == Zero) = True
(Suc n == Suc m) = n==m
(_ == _) = False
```

This is also a model for $\mathbb{N}$ but without any `null` problem.