About Your Theorem

TheoryMine

March 17, 2011

1 Recursion and Induction

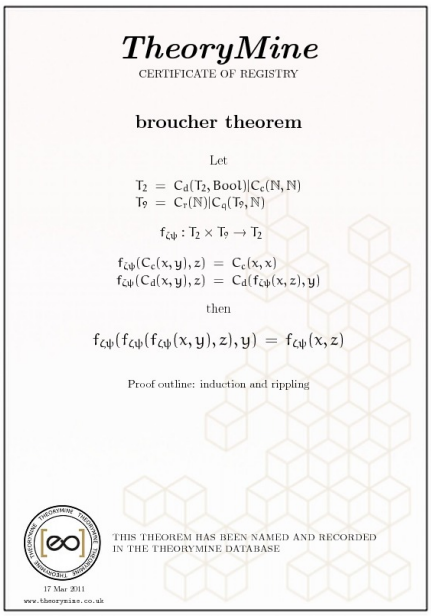

Figure 1: Your certificate

Your certificate is given in Figure 1. In order to understand it, it is first necessary to understand the concepts of *recursion* and *induction*. In this section we provide an elementary introduction to these concepts for those not already familiar with them. Those who do not need this introduction can proceed to §2.

Both new mathematical objects (see §2) and new defined functions (see §3) are defined using recursion. This is a form of definition in which the body of the definition contains the thing being defined. This sounds circular, but it need not be. Similarly, theorems about recursive objects and functions are usually proved by mathematical induction. At first sight, this form of proof can also appear to be circular, but it is not once you understand it.

1.1 The Natural Numbers

Consider, the following recursive definition of the, so called, *natural numbers*, i.e., the non-negative whole numbers $0, 1, 2, 3, \ldots$.

$$\mathsf{N} \quad = \quad 0|s(\mathsf{N}) \tag{1}$$

where s is called the successorfunction. This definition is in Backus-Naur Form[1] (BNF). N, the set of natural numbers, which is being defined, is to the left of the = sign. On the right of the = sign is the body of the definition. This consists of two cases, separated by a vertical line |. Any particular natural number corresponds to one of these cases. Note that N occurs in the second case $s(\mathsf{N})$. This is a *step case*. The other case, 0, is a *base case*.

The occurrence of a concept in the body of its own definition is an indication that the definition is *recursive*. Think of this recursive definition, not as a circle, but as a spiral. 0 is at the centre of the spiral; each application of s winds the spiral out by a complete circuit. This process generates a representation of the natural numbers of the form $0, s(0), s(s(0)), s(s(s(0))), \ldots$, where each application of s increases the number by 1. This representation is due to the mathematician Giuseppe Peano[2] (1858-1932).

0 and s are called *constructor functions*, because they are used to construct new types of mathematical objects. Note that constructor functions are *deliberately* not defined. They are taken as primitive mathematical objects and are used in the definitions of other mathematical objects and defined functions. Definitions have to stop unfolding at some point, and that point is with the constructor functions.

The boolean truth values, true and false, can be defined by a degenerate form of recursion, in which there are two base cases and no step cases: $\mathbb{B} = \mathsf{true}|\mathsf{false}$.

[1] http://en.wikipedia.org/wiki/Backus_Naur_Form
[2] http://en.wikipedia.org/wiki/Peano_axioms

We will use B and N as the basis for defining new sets of recursively defined mathematical objects, such as those explained in §2 below. They will occur as some of the inputs to constructor functions.

1.2 The Addition Function

The function + can also be defined recursively. It takes two members of N as inputs and returns one as output. It also has a base case and a step case.

$$+ \quad : \quad \mathsf{N} \times \mathsf{N} \to \mathsf{N} \tag{2}$$
$$0 + y \quad = \quad y \tag{3}$$
$$s(x) + y \quad = \quad s(x + y) \tag{4}$$

Line (2) says that + takes two natural numbers as inputs and outputs one natural number. Lines (3) and (4) constitute the recursive definition of +. The formulae to the left of the = sign are the *heads* of the definition and those to the right are the *body*. Line (3) is a base case, in which + does not occur on the right hand side. Line (4) is a step case, in which + *does* occur on the right hand side. The recursion is on x, the first input of +. x is called the *recursion variable*. Note that for each of the two cases in the definition of N there is one equation in the definition of +. Moreover, in each of these equations x is instantiated to exactly one of the cases in the definition of N, i.e., 0 and $s(x)$. This kind of recursive definition is called *structural*, as it directly corresponds to the structure of the type of objects that its recursion variable ranges over.

The occurrence of + in the body of the step case does not make the definition circular. Again, it can be seen as a spiral definition. This time the recursion spirals inwards. Starting with some particular value of x, say $s(s(o))$, the step case is applied to rewrite $s(s(0)) + y$ first to $s(s(0) + y)$ and then to $s(s(0 + y))$. Now the base case of the definition can be used to rewrite this as $s(s(y))$ and the calculation is finished. Notice how the step case is organised so that + is applied to a smaller occurrence of x in the body of the definition than in the head. N has been defined to be *well founded*, which means that a sequence of its members cannot keep getting smaller forever: sooner or later it will reach 0. This means that any calculation carried out using + will eventually stop. We can only apply the step case a finite number of times. Eventually, only the base case will be applicable and, after the base case, nothing will be applicable. To contrast functions such as + with the constructor functions, such as 0 and s, + is called a *defined function*.

1.3 The Associativity of + and its Inductive Proof

The function + is *associative*, i.e., if you add three numbers, it doesn't matter if you add the first to the sum of the other two or the sum of the first two to the third — you'll get the same answer. This fact can be formalised as the theorem:

$$u + (v + w) \quad = \quad (u + v) + w \tag{5}$$

which can be proved by *induction*.

Induction is closely related to recursion. Theorems composed of recursively defined functions applied to recursively defined objects, such as the associativity of +, are usually proved by induction. Firstly, an *induction variable* is chosen from those variables in the theorem. In this case we will choose u[3], which is the first variable on each side of the theorem.

Just like a recursive definition, an inductive proof is divided into base and step cases.

- In the base cases, the theorem is proved with the induction variable being instantiated to each of the base cases of the mathematical objects it ranges over. In our case, there is only one base case, in which u is instantiated to 0. So, we have to prove the special case of the theorem:

$$0 + (v + w) \quad = \quad (0 + v) + w$$

- In the step cases, the theorem is assumed to hold for the induction variable. This is called the *induction hypothesis*. The theorem is then proved with the induction variable being instantiated to each of the step cases of the mathematical objects it ranges over. These are called the *induction conclusions*. During the proof of an induction conclusion, we are allowed to use the induction hypothesis. This kind of induction is called *structural* because its cases correspond to the structure of the type of objects that its induction variable ranges over.

[3] See below for the effect of chosing v or w

In our case, there is only one step case. In its induction conclusion, u is instantiated to $s(u)$. So, we assume the induction hypothesis:

$$u + (v + w) = (u + v) + w \quad (6)$$

and then have to prove the induction conclusion:

$$s(u) + (v + w) = (s(u) + v) + w$$

At first sight, inductive proofs look circular. During the proof of the step case, we *assume the theorem already holds*. But, once again, the inductive proof is really spiral. We start by proving that the theorem holds for the centre of the spiral (the base case), then the step case can be used to prove that it holds for one circuit of the spiral, then for two circuits, then three, and so on for any number of circuits.

The base case can be proved by two applications of the base case of the recursive definition of $+$, $0 + y = y$. These can be used to rewrite the left-hand side of the base case into the right-hand side in two stages, as follows:

$$\begin{array}{rcll} 0 + (v + w) & = & v + w & \text{by (3)} \\ & = & (0 + v) + w & \text{by (3)} \end{array}$$

At this point the base case is proved.

In **the step case**, the induction conclusion can be proved by three applications of the step case of the recursive definition of $+$, namely $s(x) + y = s(x + y)$ and an application of the induction hypothesis. The left-hand side of the induction conclusion is rewritten the right-hand side in four stages, as follows:

$$\begin{array}{rcll} s(u) + (v + w) & = & s(u + (v + w)) & \text{by (4)} \\ & = & s((u + v) + w) & \text{by (6)} \\ & = & s(u + v) + w & \text{by (4)} \\ & = & (s(u) + v) + w & \text{by (4)} \end{array}$$

At which point the step case is proved. This completes the inductive proof of the associativity of $+$.

Sometimes, a proof requires more than just the recursive definitions of the defined functions and the induction hypothesis. It is necessary also to use previously proved theorems, perhaps purchased by another customer. When used in this way, theorems are often called *lemmas*.

To illustrate the use of lemmas in a proof, consider again the proof of the associativity of $+$, but this time using induction on v. This version of the proof requires the following two lemmas:

$$x + 0 = x \quad (7)$$
$$x + s(y) = s(x + y) \quad (8)$$

which are commuted versions of the defining equations of $+$: (3) and (4). This time the proof of the base case is:

$$\begin{array}{rcll} u + (0 + w) & = & u + w & \text{by (3)} \\ & = & (u + 0) + w & \text{by (7)} \end{array}$$

and that of the step case is:

$$\begin{array}{rcll} u + (s(v) + w) & = & u + s(v + w) & \text{by (4)} \\ & = & s(u + (v + w)) & \text{by (8)} \\ & = & s((u + v) + w) & \text{by (6)} \\ & = & (s(u + v)) + w & \text{by (4)} \\ & = & (u + s(v)) + w & \text{by (8)} \end{array}$$

Notice how both the defining equations and the lemmas are required. What is needed for the proof by induction on w?

2 Your Objects

$$\begin{array}{rcl} T_2 & = & C_d(T_2, \mathsf{Bool}) | C_c(\mathbb{N}, \mathbb{N}) \\ T_9 & = & C_r(\mathbb{N}) | C_q(T_9, \mathbb{N}) \end{array}$$

This part of the certificate contains the definitions of one or more *new* sets of mathematical object. Compare them with formula (1), the recursive definition of the natural numbers, in §1.1.

To the left of the $=$ signs are the names of the new sets of mathematical objects being defined. The TheoryMine naming convention for these new sets of objects restricts their names to T or T_n for some number n. To the right of the $=$ signs are the bodies of these definitions expressed as a number of cases separated by the $|$ symbol. Each case consists of a new constructor function with zero or more inputs. The TheoryMine naming convention for these new constructor functions restricts their names to C_l for some letter l. The inputs to these constructor functions can be B, N, the names of previously defined sets of objects and the name of the set being defined. In this last case, the constructor functions will define step cases, otherwise, they will define base cases. Just as with 0, s, `true` and `false` in §1.1, these new constructor functions are primitives that are not themselves defined.

3 Your Functions

$$f_{\zeta\psi} : T_2 \times T_9 \to T_2$$

$$\begin{aligned} f_{\zeta\psi}(C_c(x,y),z) &= C_c(x,x) \\ f_{\zeta\psi}(C_d(x,y),z) &= C_d(f_{\zeta\psi}(x,z),y) \end{aligned}$$

This part of the certificate contains the definitions of one or more *new* defined functions. Compare them with the formulae (2), (3) and (4), the recursive definition of $+$, in §1.2. The TheoryMine naming convention for these new defined functions restricts their names to f or f_λ for some Greek letter λ.

Lines of the form $f_\lambda : T_1 \times \ldots \times T_n \to T$ explain that f_λ takes n inputs of type $T_1, \ldots, T_n$ and returns one output of type T. The T and T_is can be B, N, the names of previously defined sets of objects and the names of the sets being defined in §2 above.

The remaining lines are equations that constitute the structural recursive definition of the function being defined, say f_λ. There will be at least one base case and one step case. The head (left-hand side) of each equation starts with f_λ and its inputs are either variables or constructor functions applied to variables. The TheoryMine naming convention for variables restricts their names to x, y, z and w. The same variable name may be recycled in different equations. In particular, it may stand for a different type of object in each equation. Within an equation variables with the same name *do* stand for the same object. The body (right-hand side) of each equation is a formula that can consist of variables, previously defined functions, the function being defined and constructor functions. One of the function's inputs is its *recursion variable*. There is exactly one defining equation for each of the cases of the recursive definition of the recursion variable's type.

4 Your Theorem

$$f_{\zeta\psi}(f_{\zeta\psi}(f_{\zeta\psi}(x,y),z),y) = f_{\zeta\psi}(x,z)$$

Proof outline: induction and rippling

This part of the certificate contains the statement of your theorem and a proof outline in the form of the identification of the induction variable. Your theorem can be proved by structural induction on this induction variable. There is one case for each of the cases of the recursively defined objects described in §2. In the step cases, we are allowed to assume an induction hypothesis. Compare this with theorem (5) and its proof in §1.3.