Inductive Definitions

Steffen van Bakel

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK

Autumn, 2022

Introduction

Many of the definitions and properties we are to study in computing depend heavily on induction. Not only is the majority of our definitions inductive in nature, also most of the proofs are inductive. Since the kind of induction we use is of a more general nature than just induction over numbers, we will have a close look at some of the various notions of induction, which forcefully bringing them down to mathematical induction.

There are many different approaches to definitions of induction in the literature, not all clear or consistent. We try our best here.

Induction

Being able to show a property over elements of a set by induction is based on how that set is defined. Basically, a set is defined inductively if the definition gives a way to go through the elements of the set in a systematic way, and that this way is exhaustive. Formally:

Definition 1 (Inductive Sets) We call a set V *inductively defined* or *defined by induction* when its definition is of the following shape:

(*base case*): There are elements $v_1, \ldots, v_n, \ldots$ (possibly infinitely many) that are declared to be in V.

(*construction case*): If $a_1, \ldots, a_n$ are all in V, then so is f; there can be more that one of these steps.

(*closure*): If a set W satisfies the first two clauses, then $V \subseteq W$.

The first clause declares the base elements of V, those elements of V whose membership is just stated. The second clause, also called the *inductive step*, states that f is an element of V under the condition that $a_1, \ldots, a_n$ all are as well; normally there is a syntactic relation between f and $a_1, \ldots, a_n$ and the latter appear as sub-expression inside f, but that need not be the case: it can be that f is shaped out of parts, or not related at all. (If it is the case that the a_i occur in f as sub-expressions for all the construction cases, we call this a definition using *structural induction*: the construction cases build a new element by adding structure.) The third case states that V is the smallest (in the sense of the subset relation) of all the sets satisfying the first two cases; this fact does not follow automatically from the first two, so has to be imposed.

Basically, this kind of definition implies that when you are considering an element v of V, since V is defined this way, you know that either v is one of the base cases, so is equal to one of the v_i, or it is an f and the $a_1, \ldots, a_n$ can all be assumed to be in V as well. This is exhaustive, there exist no elements in V that are not obtained this way.

This third case also makes it possible to prove properties over V using inductive proofs.

Example 2 (Proof by induction for V) Let $P(\cdot)$ be a property over elements of V, and we want to show that it holds for all elements of V. We can achieve this when defining $W = \{v \in V \mid P(v)\}$, and showing that W satisfies the first and second clause, since then by the third clause we get $V \subseteq W$, so $\forall x \in V \ (x \in W) \stackrel{\Delta}{=} \forall x \in V \ (P(x))$. So, by definition of V, to show $\forall x \in V \ (P(x))$ it is sufficient to show:

(*base case*): All elements $v_1, \ldots, v_n, \ldots$ are in W, so for all $v \in \{v_1, \ldots, v_n, \ldots\}$, $P(v)$ holds.

(*construction case*): For every step, assuming $a_1, \ldots, a_n$ are all in W, then so is f; this is equivalent to saying: assuming $P(a_i)$ holds for all $1 \leq i \leq n$, then also $P(f)$ holds.

The assumption '$P(a_i)$ holds for all $1 \leq i \leq n$' (used in the second part) is often called the *inductive hypothesis*. A common way to describe the second step in the above proof is then to say that it follows '*from the inductive hypothesis*' (abbreviated by *IH*), or '*by induction*'. Notice that, in the proof, the assumption *is needed* to show that $P(f)$; without it, there would be no way to show the result. Notice that we do not need to check that '$P(a_i)$ holds for all $1 \leq i \leq n$', but just assume it, since we are showing an implication, and therefore only need to consider the case that the assumption holds.[1]

It is important to note that every inductively defined set has its own notion of inductive proof structure.

An alternative way of defining V is using inference rules.

Definition 3 We call a set V *inductively defined using inference rules* when its definition is of the following shape:

$$(\textit{base case}): \ \frac{}{v_1 \in V} \quad \cdots \quad \frac{}{v_n \in V} \quad \cdots$$

$$(\textit{construction case}): \ \frac{a_1 \in V \quad \ldots \quad a_n \in V}{f \in V} \quad \cdots$$

We write $v \in V$ if there exists a derivation using these two steps that has this conclusion. It is implied that no elements of V exist that are not added to V this way: the derivations for establish V as the smallest set which membership satisfies these rules; this gives an inductive proof structure, but now over derivations.

For any system defined using inference rules we have the full power of induction. The *set of derivations* is structurally defined and any (suitable) property over derivable expressions is actually a property over *derivations* that can be proven by *induction on the definition of derivations*.

This definition defines V in a different way, but membership of V is exactly the same: either $v \in V$ follows by rule (*base case*) and then $v = v_i$ for some i, or it follows from rule (*construction case*) and then all for all a_i we have a sub-derivation for $a_i \in V$. But there is an important difference. Rather than defining a set, an inference system defines a set of derivations (trees structures constructed using the rules, with the base cases in the leafs), that are inductively defined. We can only say that $v \in V$ if there exists a derivation that shows this; now the objects we prove over are derivations, and a proof of $\forall x \in V \ (P(x))$ would be a proof by induction on the structure of derivations:

Example 4 (Proof by induction) Let $P(\cdot)$ be a property over elements of V, and we want to show that it holds for all elements of V: if $v \in V$, then $P(v)$ holds. We now have to look at how $v \in V$ is defined; when using rules, this statement occurs at the end of a derivation using

[1] If it doesn't, the implication always holds; the implication can only be false if the assumption holds but the conclusion does not, and we only need to check that that is not the case

the rules of Definition 3, which means that it either ends with the step:

(*base case*): We need to show for all $v_1, \ldots, v_n, \ldots$ that $P(v)$ holds.

(*construction case*): For every step, assuming P holds for all $a_1, \ldots, a_n$, show that then also $P(f)$ holds.

If these result are shown, since $v \in V$ is defined using inference rules and we have considered all the options, we have shown $\forall x \in V \ \big(P(x)\big)$.

So the structure of the proof is very much the same, but this way the inductive structure is clearer, since, in the last part, all assumptions $a_i \in V$ are the conclusion of sub-derivations of the one showing $f \in V$, and by induction we can assume P for all things derived in sub-derivations, so holds for each $a_1, \ldots, a_n$.

Example: Peano arithmetic

In 1889 Peano defined a set of axioms for the natural numbers.

Definition 5 The set $\mathbb{N}$ is defined as the set satisfying:

1) $0 \in \mathbb{N}$.

2) If $n \in \mathbb{N}$, then $Succ(n) \in \mathbb{N}$ (*Succ* is the successor function).

3) For all $n \in \mathbb{N}$, $Succ(n) \neq 0$.

4) For all $n, m \in \mathbb{N}$, if $Succ(n) = Succ(m)$, then $n = m$.

5) Let V be a set such that $0 \in V$ and, for all $n \in \mathbb{N}$, if $n \in V$ then $Succ(n) \in V$, then $\mathbb{N} \subseteq V$.

By clause (*3*) the set is not cyclic, and by clause (*4*), *Succ* is an injection.

As above, since Definition 5 is an inductive definition, we get a notion of inductive proof over naturals, which is known as *Mathematical Induction*.

Let V be a set such that $0 \in V$ and, for all $n \in \mathbb{N}$, if $n \in V$ then $Succ(n) \in V$, then $\mathbb{N} \subseteq V$.

This axiom expresses that any set V that contains 0 and is *closed under Succ* (*i.e.* if $n \in V$ then $Succ(n) \in V$) contains $\mathbb{N}$, through demanding that $\mathbb{N}$ is the *smallest set* produced by Definition 5. Writing $Succ(n)$ as $n + 1$, this corresponds to:

Lemma 6 (Mathematical Induction) *Let P be a unary predicate such that*

- *$P(0)$ is true, and*
- *for every natural number k, '$P(k)$ implies $P(k+1)$' is true.*

Then $P(n)$ is true for every natural number $n \in \mathbb{N}$: $\forall n \in \mathbb{N} \ \big(P(n)\big)$.

Proof: Let $V \triangleq \{n \in \mathbb{N} \mid P(n)\}$. We will show that $0 \in V$ and that $k \in V$ implies $Succ(k) \in V$, for any $k \in \mathbb{N}$, which gives the cases:

- Note that $P(0)$ holds by definition of P. So $0 \in V$.
- Assume that $k \in V$; by definition of V, we know that $P(k)$ holds. But then by definition of P, also $P(k+1)$ holds, so by definition of V we have $k+1 \in V$. So $k \in V$ implies $k+1 \in V$, for every $k \in \mathbb{N}$.

Then, by the principle of induction $\mathbb{N} \subseteq V$, so $\forall n \in \mathbb{N} \ \big(n \in V\big)$, so $\forall n \in \mathbb{N} \ \big(P(n)\big)$. □

Notice that the second proof step only *assumes* that $k \in V$; we have no need to show that this is true, nor do we care. We need to show that the logical implication

$$k \in V \Rightarrow k + 1 \in V$$

holds, which is true also if the antecedent is false and therefore only need to consider the case that: *if $k \in V$ is true, then so is $k + 1 \in V$.*

We can now state the principle of mathematical induction in Logic:

Definition 7 (Logical rules for Mathematical Induction)

$$P(0) \wedge \forall k \in \mathbb{N}\ \big(P(k) \Rightarrow P(k+1)\big)\ \rightarrow\ \forall n \in \mathbb{N}\ \big(P(n)\big)$$

or, in an inference system:

$$\frac{P(0) \qquad \forall k \in \mathbb{N}\ \big(P(k) \Rightarrow P(k+1)\big)}{\forall n \in \mathbb{N}\ \big(P(n)\big)}$$

We can only show this way that $\forall n \in \mathbb{N}\ \big(P(n)\big)$ holds, and and *NOT*, as is often suggested by secondary school teachers, for every $n \in \mathbb{N}$ building a proof like

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(0)} \qquad \dfrac{\dfrac{}{\forall k \in \mathbb{N}\ \big(P(k) \Rightarrow P(k+1)\big)}}{P(0) \Rightarrow P(1)}}{P(1)} \qquad \dfrac{\dfrac{}{\forall k \in \mathbb{N}\ \big(P(k) \Rightarrow P(k+1)\big)}}{P(1) \Rightarrow P(2)}}{P(2)} \qquad \dfrac{\dfrac{}{\forall k \in \mathbb{N}\ \big(P(k) \Rightarrow P(k+1)\big)}}{P(n-1) \Rightarrow P(n)}}{\vdots}}{P(n)}$$

Although this is always possible, it does not give you a proof for $\forall n \in \mathbb{N}\ \big(P(n)\big)$, since this would be an infinitely large construction, which cannot give a proof, since those are always finite.

For example, $P(10^{10^{10}})$ is shown by:

$$\dfrac{\dfrac{\dfrac{}{P(0)} \qquad \dfrac{}{\forall k \in \mathbb{N}\ \big(P(k) \Rightarrow P(k+1)\big)}}{\forall n \in \mathbb{N}\ \big(P(n)\big)}}{P(10^{10^{10}})}$$

It would be quite impossible to show $P(10^{10^{10}})$ using the method mentioned above: there has not been enough time since the Big Bang to complete this.

So, to prove a property $P(x)$ for all natural numbers it suffices to show:

(*Base case*): Prove $P(0)$.

(*Inductive Case*): For every k, using the assumption that $P(k)$ holds, prove $P(k+1)$; in other words: prove $P(k) \Rightarrow P(k+1)$.

These two proofs give you the 'right' to say that $P(n)$ holds for all n.

Therefore it turns out that mathematical induction over $\mathbb{N}$ is possible, *i.e.* an accepted proof step, purely thanks to the fact that we assume that $\mathbb{N}$ is the smallest set containing 0 and being closed under *Succ*.

There is also a notion called *complete induction*, which states:

$$\frac{P(0) \qquad \forall k \in \mathbb{N}\ \big(\forall i \leq k\ \big(P(i)\big) \Rightarrow P(k+1)\big)}{\forall n \in \mathbb{N}\ \big(P(n)\big)}$$

This corresponds to mathematical induction, in the sense that we can show for both that they hold assuming that the other does.

Related to complete induction is the notion of *well-founded or Nötherian induction*. Here we show a property over a well-founded set, a set V equipped with a well-founded order R, so that there does not exist an infinite declining chain of elements of V (where we call v greater than w if $v\,R\,w$); in other words, if $v_1\,R\,v_2\,R \cdots R\,v_k\,R\ \cdots$ is an inifinite sequence, then there exists n such that for all $m > n$, $v_n = v_m$. Then the property is proven for the smallest elements in the set, and we show that $P(w)$ holds, assuming $P(v_1), \ldots, P(v_n)$ hold for some $w\,R\,v_i$ for all $1 \leq i \leq n$. Although a perfectly valid proof technique, it is not strictly speaking a proof by

induction, since not using an inductive definition.

Example: Grammars

A common way to define a set structurally is through a *grammar*; this is a set of rules, specifying how to create elements of the sets (or sentences in a language) in steps. We will use a simplified version of the Backus-Naur form to specify grammars, a notation technique for context-free grammars.

Definition 8 (BNF GRAMMAR) *1*) A BNF grammar defines '*classes*' whose names are written in angle brackets. A BNF specification is a set of *derivation rules*, written as

$$\langle \texttt{symbol} \rangle \ ::= \ \texttt{\_\_expression\_\_}$$

where the '`__expression__`' consists of one or more sequences of symbols, separated by the vertical bar '|', indicating a (discrete) choice, the whole being a possible substitution for the symbol on the left.

2) Symbols that never appear on a left-hand side are *terminals*. On the other hand, symbols that appear on a left-hand side are *non-terminals* and are always enclosed between the brackets '$\langle$' and '$\rangle$'.

3) The '::=' means that the symbol on the left must stand for one of the choices in the expression on the right; if that contains occurrences of non-terminals, those need to be instantiated using their definitions as well.

4) Only expressions without non-terminals are considered to be '*generated by the grammar*'.

5) A grammar is *well defined* if each non-terminal can be rewritten into an expression containing just terminals using the derivation rules.

6) A grammar implicitly defines a language as the smallest set of expressions that can be generated from the grammar.

A grammar specifies how to create a set of 'sentences' of a language (of which this is the grammar).

Example 9 An example of such a grammar in BNF is:

$$\begin{aligned}
\langle \texttt{expression} \rangle \ &::= \ \langle \texttt{number} \rangle \mid \langle \texttt{expression} \rangle \langle \texttt{operation} \rangle \langle \texttt{expression} \rangle \\
\langle \texttt{operation} \rangle \ &::= \ + \mid - \mid \times \\
\langle \texttt{number} \rangle \ &::= \ \langle \texttt{digit} \rangle \mid \langle \texttt{digit} \rangle \langle \texttt{number} \rangle \\
\langle \texttt{digit} \rangle \ &::= \ 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}$$

This grammar can be used to generate a set of expressions built out of numbers, '$+$', '$-$', and '$\times$', like $\{1,\ 27+3,\ 112\times 6780,\ 7+13\times 56,\ \ldots\}$. Notice that each occurrence of $\langle$`expression`$\rangle$ is eventually replaced with an expression, and that the two occurrences of $\langle$`expression`$\rangle$ in $\langle$`expression`$\rangle + \langle$`expression`$\rangle$ need not be replaced by the same. We have four non-terminals, being `expression`, `number`, `operation`, and `digit`, and terminal symbols $+$, $-$, $\times$, 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

We can abstract a little from the actual description of numbers (but of course cannot do that when dealing with a programming language) and use an *abstract syntax*:

$$\begin{aligned}
e, f \ &::= \ n \mid (e \circ f) \\
\circ \ &::= \ + \mid - \mid \times
\end{aligned}$$

Notice that we have left the syntax of numbers unspecified.

Since languages defined through grammars have a closure condition and follow the approach of Peano's definition of $\mathbb{N}$, we have a principle of induction for every such language. This implies that when looking to prove a property for all the expressions generated by a grammar,

it suffices to show the case for all the terminal symbols, and that, assuming it holds for the sub-expressions, we can show that the property holds for the composite ones.

Example: Lambda terms and β-equality

We start with a programming language that embodies the essentials of the functional programming paradigm, Church's λ-*calculus*. It forms the basis for Haskell, and is actually part of it.

The set of λ-*terms* is constructed from an infinite, countable set of term-variables and two operations, *application* and *abstraction*. The BNF-grammar for λ-terms could be written as

$$\begin{aligned}\langle \text{lambda term} \rangle &::= \langle \text{variable} \rangle \mid (\lambda \langle \text{variable} \rangle . \langle \text{lambda term} \rangle) \mid (\langle \text{lambda term} \rangle \, \langle \text{lambda term} \rangle) \\ \langle \text{variable} \rangle &::= x \mid y \mid z \mid \cdots \mid x_1 \mid x_2 \mid \cdots\end{aligned}$$

but this has as disadvantage that we need to use ellipses to represent the set of variables. Rather, we write:

Definition 10 (λ-TERMS) We take $\mathcal{V}$ as the infinite, countable set of term-variables containing lower case characters, possibly indexed, and ranged over by x, y, z, x_1, x_2, etc. We define Λ, the set of λ-terms by the (abstract) grammar:

$$\begin{array}{rcccccc} M, N & ::= & x & \mid & (\lambda x.M) & \mid & (MN) \\ & & \textit{variable} & & \textit{abstraction} & & \textit{application} \end{array}$$

Or, in an inference system:

$$\frac{}{x \in \Lambda}\,(x \in \Lambda\text{-}\textit{vars}) \qquad \frac{M \in \Lambda}{(\lambda x.M) \in \Lambda}\,(x \in \Lambda\text{-}\textit{vars}) \qquad \frac{M \in \Lambda \qquad N \in \Lambda}{(MN) \in \Lambda}$$

Notice that in this definition the variable x is used both as an actual variable as as a meta variable.

The basic computational step is that of effecting the replacement of a bound variable in an abstraction by the parameter of the application $(\lambda x.M)N$, that runs to $M\{N/x\}$, *i.e.* M with all occurrences of x replaced by N. Although defined inductively, by following the grammatical structure of terms, the substitution $\{N/x\}$, that replaces all occurrences of x in a term by N, is assumed to take place silently, and immediately, and replaces all occurrences of x in parallel. Thereby $(xx)\{N/x\}$ and (NN) are *identical*.

The notion of computation, called β-*reduction*, is defined as a the transitive closure of the relation '$\to_\beta$' on terms that specifies that, if $M \to_\beta N$, then M executes 'in one step' to N.

Definition 11 (β-CONVERSION) *1) a)* The binary *one-step* reduction relation '$\to_\beta$' on λ-terms is defined by the β-reduction rule (we will write $M \to_\beta N$ rather than $\langle M, N \rangle \in \to_\beta$):

$$((\lambda x.M)\ N) \ \to_\beta \ M\{N/x\}$$

and the 'contextual closure' rules

$$M \to_\beta N \ \Rightarrow \ \begin{cases} (PM) \ \to_\beta \ (PN) \\ (MP) \ \to_\beta \ (NP) \\ (\lambda x.M) \ \to_\beta \ (\lambda x.N) \end{cases}$$

A term of the shape $(\lambda x.M)\ N$ is called a *reducible expression* (*redex* for short); the term $M\{N/x\}$ that is obtained by reducing this term is called a *contractum* (from 'contracting the redex') or *reduct*.

b) The relation '$\to_\beta^*$' (or '$\twoheadrightarrow_\beta$') is defined as the reflexive, transitive closure of '$\to_\beta$':

$$\begin{array}{rcl} M \to_\beta N & \Rightarrow & M \to_\beta^* N \\ M & \to_\beta^* & M \\ M \to_\beta^* N \wedge N \to_\beta^* P & \Rightarrow & M \to_\beta^* P \end{array}$$

c) '$=_\beta$' is the equivalence relation generated by '$\to_\beta^*$':

$$\begin{array}{rcl} M \to_\beta^* N & \Rightarrow & M =_\beta N \\ M =_\beta N & \Rightarrow & N =_\beta M \\ M =_\beta N \wedge N =_\beta P & \Rightarrow & M =_\beta P \end{array}$$

The relation '$\to_\beta^*$' is often written as '$\twoheadrightarrow_\beta$.'

2) We can define the relations $\to_\beta$, $\to_\beta^*$ and $=_\beta$ as sets through the rules:

$$(\to_\beta): \quad \frac{}{\langle (\lambda x.M)\ N, M\{N/x\}\rangle \in \to_\beta} \qquad \frac{\langle M,N\rangle \in \to_\beta}{\langle MP, NP\rangle \in \to_\beta} \qquad \frac{\langle M,N\rangle \in \to_\beta}{\langle PM, PN\rangle \in \to_\beta} \qquad \frac{\langle M,N\rangle \in \to_\beta}{\langle \lambda x.M, \lambda x.N\rangle \in \to_\beta}$$

$$(\to_\beta^*): \quad \frac{\langle M,N\rangle \in \to_\beta}{\langle M,N\rangle \in \to_\beta^*} \qquad \frac{}{\langle M,M\rangle \in \to_\beta^*} \qquad \frac{\langle M,N\rangle \in \to_\beta^* \quad \langle N,P\rangle \in \to_\beta^*}{\langle M,P\rangle \in \to_\beta^*}$$

$$(=_\beta): \quad \frac{\langle M,N\rangle \in \to_\beta^*}{\langle M,N\rangle \in =_\beta} \qquad \frac{\langle M,N\rangle \in =_\beta}{\langle N,M\rangle \in =_\beta} \qquad \frac{\langle M,N\rangle \in =_\beta \quad \langle N,P\rangle \in =_\beta}{\langle M,P\rangle \in =_\beta}$$

3) We can directly define the statements $M \to_\beta N$, $M \to_\beta^* N$ and $M =_\beta N$ through the rules:

$$(\to_\beta): \quad \frac{}{(\lambda x.M)\ N \to_\beta M\{N/x\}} \qquad \frac{M \to_\beta N}{MP \to_\beta NP} \qquad \frac{M \to_\beta N}{PM \to_\beta PN} \qquad \frac{M \to_\beta N}{\lambda x.M \to_\beta \lambda x.N}$$

$$(\to_\beta^*): \quad \frac{M \to_\beta N}{M \to_\beta^* N} \qquad \frac{}{M \to_\beta^* M} \qquad \frac{M \to_\beta^* N \quad N \to_\beta^* P}{M \to_\beta^* P}$$

$$(=_\beta): \quad \frac{M \to_\beta^* N}{M =_\beta N} \qquad \frac{M =_\beta N}{N =_\beta M} \qquad \frac{M =_\beta N \quad N =_\beta P}{M =_\beta P}$$

Notice that these three variants define different things: the first is a logical relation between statements, the second specify sets through derivations, and the third define inferable statements. All three give a notion of induction, however, over either statements, elements in the sets, or derivable expressions.

For example, using the first version, we can write

$$\begin{array}{rcl} (\lambda xyz.xz\ (yz))\ (\lambda ab.a) & \to_\beta & \lambda yz.(\lambda ab.a)\ z\ (yz) \\ & \to_\beta & \lambda yz.(\lambda b.z)\ (yz) \\ & \to_\beta & \lambda yz.z \end{array}$$

But using the third version (as for the second), we would need to build a derivation for each reduction step, as in

$$\frac{}{(\lambda xyz.xz\ (yz))\ (\lambda ab.a) \to_\beta \lambda yz.(\lambda ab.a)\ z\ (yz)}$$

$$\frac{\dfrac{\dfrac{\dfrac{}{(\lambda ab.a)\ z \to_\beta \lambda b.z}}{(\lambda ab.a)\ z\ (yz) \to_\beta (\lambda b.z)\ (yz)}}{\lambda z.(\lambda ab.a)\ z\ (yz) \to_\beta \lambda z.(\lambda b.z)\ (yz)}}{\lambda yz.(\lambda ab.a)\ z\ (yz) \to_\beta \lambda yz.(\lambda b.z)\ (yz)} \qquad \frac{\dfrac{\dfrac{}{(\lambda b.z)\ (yz) \to_\beta z}}{\lambda z.(\lambda b.z)\ (yz) \to_\beta \lambda z.z}}{\lambda yz.(\lambda b.z)\ (yz) \to_\beta \lambda yz.z}$$

Thereby the expression 'proof by induction on the definition of $=_\beta$' or 'proof by induction on the derivation for $=_\beta$' are both valid, but depend on in which way $=_\beta$ has been defined.

That steps like

$$M \to_\beta N \ \Rightarrow\ M \to_\beta^* N \qquad M \to_\beta^* N \ \Rightarrow\ M =_\beta N \qquad M =_\beta N \ \Rightarrow\ N =_\beta M$$

are inductive might be confusing: after all, the terms stay the same, so where is the inductive structure? Of course it is the definition of the relations that are inductive: a pair is added to

the relation because another pair is already in there. The inference rules

$$\frac{M \to_\beta N}{M \to_\beta^* N} \qquad \frac{M \to_\beta^* N}{M =_\beta N} \qquad \frac{M =_\beta N}{N =_\beta M}$$

might be clearer here since they extend an existing derivation by one step, adding structure.

Example: Simple type assignment for lambda terms

Type assignment assigns types to λ-terms, where types are defined as follows:

Definition 12 *1)* $\mathcal{T}_C$, the set of *types*, ranged over by $A, B, \ldots$, is defined over a set of *type variables* $\mathcal{V}$, ranged over by φ, by:

$$A, B \ ::= \ \varphi \mid (A \to B)$$

2) A *statement* is an expression of the form $M : A$, where $M \in \Lambda$ and $A \in \mathcal{T}_C$. M is called the *subject* and A the *predicate* of $M{:}A$.

3) A *context* Γ is a set of statements with only distinct variables as subjects; we use $\Gamma, x{:}A$ for the context defined as $\Gamma \cup \{x{:}A\}$ where either $x{:}A \in \Gamma$ or x does not occur in Γ, and $x{:}A$ for $\emptyset, x{:}A$. We write $x \in \Gamma$ if there exists A such that $x{:}A \in \Gamma$, and $x \notin \Gamma$ if this is not the case.

The notion of context will be used to collect all statements used for the free variables of a term when typing that term. In the notation of types, right-most and outer-most parentheses are normally omitted, so $(\varphi_1 \to \varphi_2) \to \varphi_3 \to \varphi_4$ stands for $((\varphi_1 \to \varphi_2) \to (\varphi_3 \to \varphi_4))$.

Definition 13 *1) Derivable judgements* are (inductively) defined by

- For every context Γ, variable x and type A, $\Gamma, x{:}A \vdash_C x : A$ (holds, is true).
- For every context Γ, variable x, term M and types A and B, if $\Gamma, x{:}A \vdash_C M : B$, then $\Gamma \vdash_C \lambda x.M : A \to B$.
- For every context Γ, variable x, terms M and N, and types A and B, if $\Gamma \vdash_C M : A \to B$ and $\Gamma \vdash_C N : A$, then $\Gamma \vdash_C MN : B$.

2) The set of *valid type judgements* (VTJ) is (inductively) defined by

- For every context Γ, variable x and type A, $\Gamma, x{:}A \vdash_C x : A$ is a VTJ.
- For every context Γ, variable x, term M and types A and B, if $\Gamma, x{:}A \vdash_C M : B$ is a VTJ, then so is $\Gamma \vdash_C \lambda x.M : A \to B$.
- For every context Γ, variable x, terms M and N, and types A and B, if $\Gamma \vdash_C M : A \to B$ and $\Gamma \vdash_C N : A$ are VTJ, then so is $\Gamma \vdash_C MN : B$.

Notice that this is an inductive definition that is not structural.

3) *Type assignment derivations* are defined by the following inference rules.

$$(Ax): \ \frac{}{\Gamma, x{:}A \vdash x : A} \qquad (\to I): \ \frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}\ (x \notin \Gamma) \qquad (\to E): \ \frac{\Gamma \vdash P : A \to B \qquad \Gamma \vdash Q : A}{\Gamma \vdash PQ : B}$$

We will write $\Gamma \vdash_C M : A$ if this statement is derivable, i.e. if there exists a derivation, built using these three rules, that has this statement in the bottom line.

These three definitions can be seen as defining the same thing, but this is only in appearance. The first alternative defines statements, which validity can depends assuming that other statements hold, the second defines a ternary relation, a set, and the third defines derivations. Thereby proofs over these definitions, though all by induction, are structured differently.