Lecture Notes on Natural Numbers

15-317: Constructive Logic
Frank Pfenning*

Lecture 11
Tuesday, February 21, 2023

1 Introduction

In this lecture we discuss the data type of natural numbers. They serve as a prototype for a variety of inductively defined data types, such as lists or trees. Together with quantification coming up in Lecture 13, this allow us to reason constructively about natural numbers and extract corresponding functions. The constructive system for reasoning logically about natural numbers is called *intuitionistic arithmetic* or *Heyting arithmetic* Heyting [1934, 1956]. The *classical* version of the same principles is called *Peano arithmetic* Peano [1889]. Both of these are usually introduced *axiomatically* rather than as an extension of natural deduction as we do here. We will check some local properties (reduction and expansion), but the global ones are rather complex. The consistency of arithmetic was proven by Gentzen [1936] using an analysis of the structure of derivations using ordinals, which goes beyond the scope of this course.

2 Induction

It is somewhat unintuitive to think of nat as a proposition. But remember that the meaning of a proposition is determined by what counts as a verification of it, so we can analyze the verifications of nat to understand it. Not surprisingly, the verifications of nat will exhibit the structure of natural numbers as defined from zero and a successor function.

An equally sound and more intuitive approach is to think of nat as a *type*, especially if we remember the correspondence between propositions and types. If we start from this point of view, the best judgment is $M : A$ rather than A *true* because it makes the *inhabitant* M of a type A explicit.

Under this view, there are two introduction rules, one for zero and one for successor.

$$\frac{}{\mathsf{0} : \mathsf{nat}}\ \mathsf{nat}I_0 \qquad \frac{M : \mathsf{nat}}{\mathsf{s}\,M : \mathsf{nat}}\ \mathsf{nat}I_s$$

*Contributions by André Platzer

Intuitively, these rules express that $\mathsf{0}$ is a natural number ($\mathsf{nat}I_0$) and that the successor $\mathsf{s}\,n$ is a natural number if n is a natural number. This definition has a different character from the previous definitions. For example, we defined the meaning of $A \wedge B$ *true* from the meanings of A *true* and the meaning of B *true*, all of which are propositions. It is even different from the proof term assignment rules where, for example, we defined $\langle M, N\rangle : A \wedge B$ in terms of $M : A$ and $N : B$. In each case, the proposition is decomposed into its parts.

Here, the types in the conclusion and premise of the $\mathsf{nat}I_s$ rules are the same, namely nat. Fortunately, the *term* M in the premise is a part of the term $\mathsf{s}\,M$ in the conclusion, so the definition is not circular, because the judgment in the premise is still smaller than the judgment in the conclusion.

In (verificationist) constructive logic truth is defined by the introduction rules. The resulting implicit principle, that nothing is true unless the introduction rules prove it to be true, is of deep significance here. Nothing else is a natural number, except the objects constructed via $\mathsf{nat}I_s$ from $\mathsf{nat}I_0$. The rational number $\frac{7}{4}$ cannot sneak in claiming to be a natural number (which, by $\mathsf{nat}I_s$ would also make its successor $\frac{11}{4}$ claim to be natural).

But what should the elimination rule be? We cannot decompose the proposition into its parts, so we decompose the term instead. Natural numbers have two introduction rules just like disjunctions. Their elimination rule, thus, also proceeds by cases, accounting for the possibility that a given n of type nat is either $\mathsf{0}$ or $\mathsf{s}\,x$ for some x. A property $C(n)$ is true if it holds no matter whether the natural number n was introduced by $\mathsf{nat}I_0$ so is zero or was introduced by $\mathsf{nat}I_s$ so is a successor.

$$\frac{n : \mathsf{nat} \quad C(\mathsf{0})\ \mathit{true} \qquad \dfrac{\dfrac{}{x : \mathsf{nat}}\,x \quad \dfrac{}{C(x)\ \mathit{true}}\,u}{\vdots}\ C(\mathsf{s}\,x)\ \mathit{true}}{C(n)\ \mathit{true}}\ \mathsf{nat}E^{x,u}$$

In words: In order to prove property C of a natural number n we have to prove $C(\mathsf{0})$ and also $C(\mathsf{s}\,x)$ under the assumption that $C(x)$ for a new parameter x. The scope of x and u is just the rightmost premise of the rule. This corresponds exactly to proof by induction, where the proof of $C(\mathsf{0})$ is the base case, and the proof of $C(\mathsf{s}\,x)$ from the assumption $C(x)$ is the induction step. That is why $\mathsf{nat}E^{x,u}$ is also called an *induction rule* for nat.

We should immediately check local soundness and completeness, but let's postpone until Sections 6 and 10. But let's notice a few other things about this rule.

- The rules mixes the judgment $M : A$ with the judgment C *true* so it looks more like a traditional rule of induction. We can equally well use the judgment $N : C$ instead, which we will do later in this lecture.
- The proposition C can mention terms—natural numbers in this case. We write $C(x)$ for a proposition with a free variable x and $C(M)$ for substituting a term M for x in C.
- We managed to state this rule without any explicit appeal to universal quantification, using parametric judgments instead. We could, however, write it down with explicit

quantification, in which case it becomes:

$$\forall n{:}\mathsf{nat}.\ C(\mathsf{0}) \supset (\forall x{:}\mathsf{nat}.\ C(x) \supset C(\mathsf{s}\,x)) \supset C(n)$$

for an arbitrary property C of natural numbers. It is an easy exercise to prove this with the induction rule above, since the respective introduction rules lead to a proof that exactly has the shape of $\mathsf{nat}E^{x,u}$.

In the next section we will introduce equality of natural numbers as a form of proposition we can reason about.

3 Equality on Natural Numbers

Generally speaking, arithmetic (whether classical or intuitionistic) uses a primitive notion of *equality on natural numbers* that is defined by a collection of axioms.

There are many ways to define and reason with equality. The one we choose here is the one embedded in arithmetic where we are only concerned with numbers. Thus we are trying to define $n = k$ only for natural numbers n and k. Of course, $n = k$ must be a *proposition*, not a term. As a proposition, we will use the techniques of the course and define it by means of introduction and elimination rules!

The introduction rules are straightforward.[1]

$$\frac{}{\mathsf{0} = \mathsf{0}\ \mathit{true}}\ {=}I_{00} \qquad \frac{n = k\ \mathit{true}}{\mathsf{s}\,n = \mathsf{s}\,k\ \mathit{true}}\ {=}I_{ss}$$

If we take this as our definition of equality on natural numbers, how can we use the knowledge that $n = k$? If n and k are both zero, we cannot learn anything. If both are successors, we know their argument must be equal. Finally, if one is a successor and the other zero, then this is contradictory and we can derive anything.

$$\text{no rule } E_{00} \qquad \frac{\mathsf{0} = \mathsf{s}\,k\ \mathit{true}}{C\ \mathit{true}}\ {=}E_{0s} \qquad \frac{\mathsf{s}\,n = \mathsf{0}\ \mathit{true}}{C\ \mathit{true}}\ {=}E_{s0} \qquad \frac{\mathsf{s}\,n = \mathsf{s}\,k\ \mathit{true}}{n = k\ \mathit{true}}\ {=}E_{ss}$$

Local soundness is very easy to check, but what about local completeness? It turns out to be a complicated issue so we will not discuss it here.

4 Equality is Reflexive

As a simple inductive theorem we now present the reflexivity of equality.

Theorem 1 *Given* n : nat. *Then* $n = n$.

Proof: By induction on n.

[1] As a student observed in lecture, we could also just state $x = x\ \mathit{true}$ as an inference rule with no premise. However, it is difficult to justify the elimination rules we need for Heyting arithmetic from this definition.

Base: $n = 0$. Then $0 = 0$ by rule $=I_{00}$

Step: Assume $x = x$ for an arbitrary natural numbers x. We have to show $\mathsf{s}\,x = \mathsf{s}\,x$, which follows by $=I_{ss}$.

□

This proof is small enough so we can present it in the form of a natural deduction. For the induction, we use $C(x) = (x = x)$.

$$\dfrac{n : \mathsf{nat} \quad \dfrac{}{0 = 0\ \mathit{true}}\,{=}I_{00} \quad \dfrac{\dfrac{}{x = x\ \mathit{true}}\,u}{\mathsf{s}\,x = \mathsf{s}\,x\ \mathit{true}}\,{=}I_{ss}}{n = n\ \mathit{true}}\,\mathsf{nat}E^{x,u}$$

Note how in the first branch we have to prove $C(0) = (0 = 0)$ and in the second branch we have to prove $C(\mathsf{s}\,x) = (\mathsf{s}\,x = \mathsf{s}\,x)$ assuming $C(x) = (x = x)$.

The hypothesis $x : \mathsf{nat}$ introduced by $\mathsf{nat}E^{x,u}$ is implicitly used to establish that $x = x$ is a well-formed proposition, but is otherwise not explicit in the proof.

Now we can define a derived rule of inference:

$$\dfrac{\overset{n\ :\ \mathsf{nat}}{\cdots\cdots\cdots}}{n = n\ \mathsf{true}}\ \mathsf{refl}$$

by using what we just proved. We usually suppress the premise $n : \mathsf{nat}$ since we already must know $n : \mathsf{nat}$ for the proposition $n = n$ to be well-formed.

5 Decidability of Equality

Another fundamental property that is important in Heyting arithmetic is the *decidability of equality*.

Theorem 2 (Decidability of Equality) $\forall n{:}\mathsf{nat}.\, \forall k{:}\mathsf{nat}.\, n = k \vee \neg(n = k)$.

Proof: Let's carry out an induction proof in a mathematical style. From there is could be rewritten in the form of natural deduction, but since we are used to this it shouldn't be difficult to do so. In this proof we write $\neg(a = b)$ as $a \neq b$. Something we missed in lecture is that we need an inner quantifier on k, which we don't formally introduce until Lecture 13. So let's stick with the mathematical presentation for this lecture.

The proof is by induction on n. We have hypotheses $n : \mathsf{nat}$ and have to prove $\forall k.\, n = k \vee n \neq k$. Our proposition $C(x)$ is therefore $\forall k.\, x = k \vee x \neq k$ and our first inference (working bottom-up) is $\mathsf{nat}E^{x,u}$, with two cases.

Case: $n = 0$. Then we have to show $\forall k.\, 0 = k \vee 0 \neq k$. This means we have to prove, for an arbitrary $k : \mathsf{nat}$ that either $0 = k \vee 0 \neq k$. To do that we apply $\mathsf{nat}E^{y,w}$ on $k : \mathsf{nat}$ with two further branches.

Subcase: $k = 0$. Then $0 = 0$ by $=I_{00}$.

Subcase: $k = \mathsf{s}\,y$ for an arbitrary $y : \mathsf{nat}$. Then we have to prove $0 \neq \mathsf{s}\,y$, so we assume $0 = \mathsf{s}\,y$ and derive a contradiction by rule $=E_{0s}$.

This use of $\mathsf{nat}E$ did not require a use of the induction hypothesis w.

Case: $n = \mathsf{s}\,x$. We have to show that, under the hypothesis $\forall k.\, x = k \vee x \neq k$, we can conclude that

$$\forall k.\, \mathsf{s}\,x = k \vee \mathsf{s}\,x \neq k$$

As in the previous case, we now apply $\mathsf{nat}E^{y,w}$ to the hypothesis $k : \mathsf{nat}$ and obtain the following two subcases.

Subcase: $k = 0$. Then $\mathsf{s}\,x \neq 0$ by implication introduction and $=E_{s0}$.

Subcase: $k = \mathsf{s}\,y$. Then we instantiate the induction hypothesis $\forall k.\, x = k \vee x \neq k$ with y to conclude $x = y \vee x \neq y$. We distinguish two more subcases.

Subsubcase: $x = y$. Then $\mathsf{s}\,x = \mathsf{s}\,y$ by $=I_{ss}$.

Subsubcase: $x \neq y$. Then $\mathsf{s}\,x \neq \mathsf{s}\,y$ by implication introduction, $=E_{ss}$ (to get $x = y$), and implication elimination with $x \neq y$.

□

6 Local Proof Reduction

We would like to check that the rules for natural numbers are locally sound and complete. For soundness, we verify that no matter how we introduce the judgment $n : \mathsf{nat}$, we can find a "more direct" proof of the conclusion. In the case of $\mathsf{nat}I_0$ this is easy to see, because the second premise already establishes our conclusion directly.

$$\dfrac{\dfrac{}{0 : \mathsf{nat}}\,\mathsf{nat}I_0 \quad \mathcal{E}\;\; C(0)\ \mathit{true} \qquad \dfrac{\dfrac{}{x : \mathsf{nat}}\,x \quad \dfrac{}{C(x)\ \mathit{true}}\,u \qquad \mathcal{F}}{C(\mathsf{s}\,x)\ \mathit{true}}}{C(0)\ \mathit{true}}\,\mathsf{nat}E^{x,u} \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{E}\\ C(0)\ \mathit{true}\end{array}$$

The case where $n = \mathsf{s}\,n'$ is more difficult and more subtle. Intuitively, we should be using the deduction of the second premise for this case.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ n' : \mathsf{nat}\end{array}}{\mathsf{s}\,n' : \mathsf{nat}}\,\mathsf{nat}I_s \qquad \begin{array}{c}\mathcal{E}\\ C(0)\ true\end{array} \qquad \dfrac{\overline{x : \mathsf{nat}} \quad \overline{C(x)\ true}^{\,u}}{\begin{array}{c}\mathcal{F}\\ C(\mathsf{s}\,x)\ true\end{array}}}{C(\mathsf{s}\,n')\ true}\,\mathsf{nat}E^{x,u}$$

$$\Longrightarrow_R \quad \dfrac{\begin{array}{c}\mathcal{D}\\ n' : \mathsf{nat}\end{array} \qquad \dfrac{\begin{array}{c}\mathcal{D}\\ n' : \mathsf{nat}\end{array} \quad \begin{array}{c}\mathcal{E}\\ C(0)\ true\end{array} \quad \dfrac{\overline{x : \mathsf{nat}} \quad \overline{C(x)\ true}^{\,u}}{\begin{array}{c}\mathcal{F}\\ C(s\,x)\ true\end{array}}}{C(n')\ true}\,\mathsf{nat}E^{x,u}}{\begin{array}{c}[n'/x]\mathcal{F}'\\ C(s\,n')\ true\end{array}}$$

It is difficult to see in which way this is a reduction: $\mathcal{D}$ is duplicated, $\mathcal{E}$ persists, and we still have an application of $\mathsf{nat}E$. The key is that the term we are eliminating with the applicaton of $\mathsf{nat}E$ becomes smaller: from $\mathsf{s}\,n'$ to n'. In hindsight we should have expected this, because the term is also the only component getting smaller in the second introduction rule for natural numbers. Fortunately, the term that $\mathsf{nat}E$ is applied to can only get smaller finitely often because it will ultimately just be 0, so will be back in the first local reduction case.

The computational content of this reduction is more easily seen in a different context, so we move on to discuss primitive recursion.

The question of local expansion is trickier and postponed to Section 10 which we did not cover in lecture.

7 Primitive Recursion

Reconsidering the elimination rule for natural numbers, we can notice that we exploit the knowledge that $n : \mathsf{nat}$, but we only do so when we are trying to establish the truth of a proposition, $C(n)$. However, we are equally justified in using $n : \mathsf{nat}$ when we are trying to establish a typing judgment of the form $M : A$. The rule, also called *rule of primitive recursion* for nat, then becomes

$$\dfrac{n : A \quad M_0 : A \qquad \begin{array}{c}\overline{x : \mathsf{nat}} \quad \overline{r : A}\\ \vdots\\ M_s : A\end{array}}{R(n, M_0, x.\, r.\, M_s) : A}\,\mathsf{nat}E^{x,r}$$

Here, R is a new term constructor,[2] the term M_0 is the zero case where $n = 0$, and the term M_s captures the successor case where $n = \mathsf{s}\,n'$. In the latter case x is a new parameter

[2] R suggests recursion

introduced in the rule that stands for n'. And r stands for the result of R when applied to n', which corresponds to an appeal to the induction hypothesis. The notation $x.\, r.\, M_s$ indicates that occurrences of x and r in M_s are bound with scope M_s. The fact that both are bound corresponds to the assumptions $x : \mathsf{nat}$ and $r : A$ that are introduced to prove $t_s : A$ in the rightmost premise.

The local reduction rules may help explain this. We first write then down just on the terms, where they are computation rules.

$$\begin{array}{lcl} R(\mathsf{0}, M_0, x.\, r.\, M_s) & \Longrightarrow_R & M_0 \\ R(\mathsf{s}\, n', M_0, x.\, r.\, M_s) & \Longrightarrow_R & [R(n', M_0, x.\, r.\, M_s)/r][n'/x]\, M_s \end{array}$$

The second case reduces to the term M_s with parameter x instantiated to the number n' of the inductive hypothesis and parameter r instantiated to the value of R at n'. So the argument M_0 of R indicates the output to use for $n = \mathsf{0}$ and M_s indicates the output to use for $n = \mathsf{s}\, x$ as a function of the smaller number x and of r for the recursive outcome of $R(n, M_0, x.\, r.\, M_s)$.

These are still quite unwieldy, so we consider a more readable schematic form, called the *schema of primitive recursion*. If we define f by cases

$$\begin{array}{lcl} f(\mathsf{0}) & = & M_0 \\ f(s\, x) & = & M_s(x, f(x)) \end{array}$$

where the only occurence of f on the right-hand side is applied to x, then we could have defined f explicitly with

$$f = \lambda n.\, R(n, M_0, x.\, r.\, M_s(x, r))$$

To verify this, apply f to $\mathsf{0}$ and apply the reduction rules and also apply f to $\mathsf{s}\, n$ for an arbitrary n and once again apply the reduction rules.

$$\begin{array}{rcl} f(\mathsf{0}) & \Longrightarrow_R & R(\mathsf{0}, M_0, x.\, r.\, M_s(x, r)) \\ & \Longrightarrow_R & M_0 \end{array}$$

noting that the x in $x.r.M_s(\ldots)$ is not a free occurrence (indicated by the presence of the dot in x.) since it corresponds to the hypothesis $x : \mathsf{nat}$ in $\mathsf{nat}E^{x,r}$. Finally

$$\begin{array}{rcl} f(\mathsf{s}\, n) & \Longrightarrow_R & R(\mathsf{s}\, n, M_0, x.\, r.\, M_s(x, r)) \\ & \Longrightarrow_R & M_s(n, R(n, M_0, x.\, r.\, M_s(x, r))) \\ & = & M_s(n, f(n)) \end{array}$$

The last equality is justified by a (meta-level) induction hypothesis, because we are trying to show that $f(n) = R(n, M_0, x.\, r.\, M_s(x, r))$

Again, we emphasize that we go freely back and forth between propositions and types, using what is most convenient. Here it is easier to read $\mathsf{nat} \to (\mathsf{nat} \to \mathsf{nat})$ as opposed to $\mathsf{nat} \supset (\mathsf{nat} \supset \mathsf{nat})$.

Now we can define double via the schema of primitive recursion.

$$\begin{array}{lcl} \mathsf{double}(\mathsf{0}) & = & \mathsf{0} \\ \mathsf{double}(\mathsf{s}\, x) & = & \mathsf{s}\, (\mathsf{s}\, (\mathsf{double}\, x)) \end{array}$$

We can read off the closed-form definition if we wish:

$$\mathsf{double} = \lambda n.\ R(n, \mathsf{0}, x.\, r.\, \mathsf{s}\,(\mathsf{s}\,r))$$

After having understood this, we will be content with using the schema of primitive recursion. We define addition and multiplication as exercises.

$$\begin{array}{lcl} \mathsf{plus}(\mathsf{0}) & = & \lambda y.\ y \\ \mathsf{plus}(\mathsf{s}\,x) & = & \lambda y.\ \mathsf{s}\,((\mathsf{plus}\,x)\,y) \end{array}$$

Notice that plus is a function of type $\mathsf{nat} \to (\mathsf{nat} \to \mathsf{nat})$ that is primitive recursive in its (first and only) argument.

$$\begin{array}{lcl} \mathsf{times}(\mathsf{0}) & = & \lambda y.\ \mathsf{0} \\ \mathsf{times}(\mathsf{s}\,x) & = & \lambda y.\ (\mathsf{plus}\ ((\mathsf{times}\,x)\,y))\,y \end{array}$$

Sometimes slow-growing functions are actually more difficult to represent with primitive recursion. The predecessor, though, it easy because we can mention it directly on the right-hand side:

$$\begin{array}{lcl} \mathsf{pred}(\mathsf{0}) & = & 0 \\ \mathsf{pred}(\mathsf{s}\,x) & = & x \end{array}$$

or:

$$\mathsf{pred} = \lambda x{:}\mathsf{nat}.\ R(x, \mathsf{0}, x.\, r.\, x)$$

On the other hand, it is not so obvious how to turn the "half" function into the form of a primitive recursion. We can specify it easily with a set of equations, though.

$$\begin{array}{lcl} \mathsf{half}(\mathsf{0}) & = & 0 \\ \mathsf{half}(\mathsf{s}\,\mathsf{0}) & = & 0 \\ \mathsf{half}(\mathsf{s}\,\mathsf{s}\,x) & = & \mathsf{s}\,(\mathsf{half}(x)) \end{array}$$

Give it a shot!

8 Proof Terms for Induction

With proof terms for primitive recursion in place, we can revisit and make a consistent proof term assignment for the elimination form with respect to the truth of propositions, and it is the exact same.

$$\frac{\begin{array}{ccc} & \overline{x : \mathsf{nat}}\ x & \overline{u : C(x)}\ u \\ & \vdots & \\ n : \mathsf{nat} \quad M_0 : C(\mathsf{0}) & & M_s : C(\mathsf{s}\,x) \end{array}}{R(n, M_0, x.\, u.\, M_s) : C(n)}\ \mathsf{nat}E^{x,u}$$

The local reductions we discussed before for terms representing data, also work for these proofs terms.

$$\begin{array}{lcl} R(\mathsf{0}, M_0, x.\, u.\, M_s) & \Longrightarrow_R & M_0 \\ R(\mathsf{s}\, n', M_0, x.\, u.\, M_s) & \Longrightarrow_R & [R(n', M_0, x.\, u.\, M_s)/u][n'/x]\, M_s \end{array}$$

We can conclude that proofs by induction correspond to functions defined by primitive recursion, and that they compute in the same way.

9 Decidability of Equality Revisited

Now we can look back at our earlier proof and write the corresponding proof term, even without writing out a full natural deduction! To write them out in full detail requires the proof terms for the equality rules, so let's restate them here with proof terms.

$$\frac{}{\mathsf{eqI00} : \mathsf{0} = \mathsf{0}}\, =I_{00} \qquad \frac{M : n = k}{\mathsf{eqIss}\, M : \mathsf{s}\, n = \mathsf{s}\, k}\, =I_{ss}$$

$$\text{no rule } E_{00} \qquad \frac{M : \mathsf{0} = \mathsf{s}\, k}{\mathsf{eqE0s}\, M : C}\, =E_{0s} \qquad \frac{M : \mathsf{s}\, n = \mathsf{0}}{\mathsf{eqEs0} : C}\, =E_{s0} \qquad \frac{M : \mathsf{s}\, n = \mathsf{s}\, k}{\mathsf{eqEss}\, M : n = k}\, =E_{ss}$$

Using these proof terms we can write out the (almost) primitive recursive form of the function decide:

$$\begin{array}{lcl} \mathsf{decide}\, \mathsf{0}\, \mathsf{0} & = & \mathbf{inl}\, \mathsf{eqI00} \\ \mathsf{decide}\, \mathsf{0}\, (\mathsf{s}\, y) & = & \mathbf{inr}\, (\lambda u.\, \mathsf{eqE0s}\, u) \\ \mathsf{decide}\, (\mathsf{s}\, x)\, \mathsf{0} & = & \mathbf{inr}\, (\lambda u.\, \mathsf{eqEs0}\, u) \\ \mathsf{decide}\, (\mathsf{s}\, x)\, (\mathsf{s}\, y) & = & \mathbf{case}(\mathsf{decide}\, x\, y, u.\, \mathbf{inl}\, (\mathsf{eqIss}\, u), u.\, \mathbf{inr}\, (\lambda w.\, u\, (\mathsf{eqEss}\, w))) \end{array}$$

As before, we can create a λ-abstraction for the second argument, but we must also have a case distinction over it. That's a primitive recursion without a use of the recursive call, or an induction without an appeal the induction hypothesis. The only appeal to the induction hypothesis here is $\mathsf{decide}\, x$ which is then applied to y.

We can also write this out using the R notation, but there doesn't seem to be much point. But we can erase all proofs of equality and disequality and just replace them with unit. Then we obtain:

$$\begin{array}{lcl} \mathsf{decide}\, \mathsf{0}\, \mathsf{0} & = & \mathbf{inl}\, \langle\rangle \\ \mathsf{decide}\, \mathsf{0}\, (\mathsf{s}\, y) & = & \mathbf{inr}\, \langle\rangle \\ \mathsf{decide}\, (\mathsf{s}\, x)\, \mathsf{0} & = & \mathbf{inr}\, \langle\rangle \\ \mathsf{decide}\, (\mathsf{s}\, x)\, (\mathsf{s}\, y) & = & \mathbf{case}(\mathsf{decide}\, x\, y, u.\, \mathbf{inl}\, \langle\rangle, u.\, \mathbf{inr}\, \langle\rangle) \end{array}$$

At this point we have a function we might write in SML, except we would probably replace $\mathbf{inl}\langle\rangle$ with true and $\mathbf{inr}\langle\rangle$ with false. The original version above produces a proof term, but the version below erases the "proof" aspect, replacing the equality just with truth. We have already carried out this kind of erasure several times in this course.

10 Local Expansion[3]

Using primitive recursion, we can now write a local expansion.

$$\frac{\mathcal{D}}{n : \mathsf{nat}} \Longrightarrow_E \frac{\dfrac{\mathcal{D}}{n : \mathsf{nat}} \quad \dfrac{}{\mathsf{0} : \mathsf{nat}}\,\mathsf{nat}I_0 \quad \dfrac{\overline{x : \mathsf{nat}}\,x}{\mathsf{s}\,x : \mathsf{nat}}\,\mathsf{nat}I_s}{R(n, \mathsf{0}, x.\, r.\, \mathsf{s}\,x) : \mathsf{nat}}\,\mathsf{nat}E^{x,r}$$

Perhaps surprisingly, there is another option which uses the recursive result.

$$\frac{\mathcal{D}}{n : \mathsf{nat}} \Longrightarrow_E \frac{\dfrac{\mathcal{D}}{n : \mathsf{nat}} \quad \dfrac{}{\mathsf{0} : \mathsf{nat}}\,\mathsf{nat}I_0 \quad \dfrac{\overline{r : \mathsf{nat}}\,r}{\mathsf{s}\,r : \mathsf{nat}}\,\mathsf{nat}I_s}{R(n, \mathsf{0}, x.\, r.\, \mathsf{s}\,r) : \mathsf{nat}}\,\mathsf{nat}E^{x,r}$$

One of these expands into a simply proof by cases, the other into a primitive recursion.

$$\begin{aligned} \mathsf{id}_1\,\mathsf{0} &= \mathsf{0} \\ \mathsf{id}_1\,(\mathsf{s}\,x) &= \mathsf{s}\,x \end{aligned}$$

$$\begin{aligned} \mathsf{id}_2\,\mathsf{0} &= \mathsf{0} \\ \mathsf{id}_2\,(\mathsf{s}\,x) &= \mathsf{s}\,(\mathsf{id}_2\,x) \end{aligned}$$

It seems the second somehow more accurately reflects the structure of introduction rules, but it is difficult to formulate a precise criterion that would prefer the second version over the first.

References

Gerhard Gentzen. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Zeitschrift*, 112:493–565, 1936. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, 1969.

A. Heyting. *Mathematische Grundlagenforschung. Intuitionismus. Beweistheorie*. Springer, Berlin, 1934.

Arend Heyting. *Intuitionism: An Introduction*. North-Holland Publishing, Amsterdam, 1956. 3rd edition, 1971.

Giuseppe Peano. *Arithmetices Principia, Nova Methodo Exposita*. Fratres Bocca, 1889.

[3] not covered in lecture