These are *rough notes* based on my own notes I prepared for the Week 5 lecture. They go with the annotated lecture slides you can see on the "More" page of the course website. Although they are rough, I'm providing them here in case they help you remember what we did in lecture.

(In case you're curious: I don't follow these notes exactly in lecture. Once I'm in class, I try not to read them at all if I can avoid it. So why do I write them? Because writing these notes is like a practice run for the lecture. I find any presentation I give (like a lecture) goes a lot more smoothly if I've spent a bit of time thinking about every part of it.)

1 Legend

Here's how to read these notes:

Text that was already on the slides is in a box, like this.
Descriptions of things that were already on the slides are formatted like this.

> Text I planned to write during lecture is formatted like this, with a line to the left.
>
> *Descriptions of other things I planned to draw or write or do are formatted like this.*

Anything else is text I planned to say, or things I planned to ask.

2 Complete Induction (continued)

2.1 Proof template

Complete induction proof template

(Define some predicate $P : \mathbb{N} \to \{\mathrm{T}, \mathrm{F}\}$.)

Let $n \in \mathbb{N}$ be arbitrary.
Assume $\forall k \in \mathbb{N}.\ ((k < n)$ IMPLIES $P(k))$. (Induction hypothesis)
... various cases ...
$P(n)$
$\big[\forall k \in \mathbb{N}.\ ((k < n)$ IMPLIES $P(k))\big]$ IMPLIES $P(n)$ (optional)
$\forall n \in \mathbb{N}.\ \big[\big[\forall k \in \mathbb{N}.\ ((k < n)$ IMPLIES $P(k))\big]$ IMPLIES $P(n)\big]$ (optional)
$\forall n \in \mathbb{N}.\ P(n)$ by complete induction

2.2 Example proof

Let's try using this to prove something.

Repeat the proof template, excluding "define some predicate" and the optional lines.
Theorem 1. Every integer greater than 1 is a product of prime numbers.

This is part of the fundamental theorem of arithmetic. The full theorem says there's only one way to write the product.

> Proof by complete induction:
>
> For $n \in$, let $P(n) =$

Ask what we should put here. There's more than one right answer.

> *(Fill it in as:)* For $n \in \mathbb{N}$, let $P(n) =$ "$(n > 1)$ IMPLIES (n is a product of primes)".
>
> Let $n \in \mathbb{N}$. Suppose $\forall k \in \mathbb{N}.\ ((k < n)$ IMPLIES $P(n))$. (I.H.)

Ask for help writing the rest.

> Case 1: $n \leq 1$. Then $P(n)$ vacuously true.
>
> Case 2: n is prime. Then it's a product of 1 prime, so $P(n)$.
>
> Case 3: $n > 1$ and n is not prime.
>
> That means $\exists a \in \mathbb{N}.\ \exists b \in \mathbb{N}.\ (a > 1$ AND $b > 1$ AND $n = ab)$.
> $a = \frac{n}{b}$, so $a < n$, so by I.H, $P(a)$: i.e. $(a > 1)$ IMPLIES $\cdots$.
> $a > 1$, so by modus ponens, a is a product of primes.
> So is b (same reason).
> So $n = ab$ is a product of all of those primes together.
> So $P(n)$.
>
> In all cases, $P(n)$.
>
> By complete induction, $\forall n \in \mathbb{N}.\ P(n)$.

2.3 Complete vs. simple induction

(side by side)

Simple Induction
Base case: $P(0)$
Induction step:
Let $n \in \mathbb{N}$; assume $P(n)$.
$\vdots$
$P(n+1)$

Complete Induction
Let $n \in \mathbb{N}$; assume $P(0), \ldots, P(n-1)$.
$\vdots$
$P(n)$

Draw an arrow from Simple to Complete. Easy to convert.

Draw an arrow from Complete to Simple. Done in tutorial (also on website).

3 Recursively defined sets

Recursively defined sets

Finite bit strings:

Define $\{0,1\}^*$ by:

Base case: $\lambda \in \{0,1\}^*$ *(arrow to λ)* Empty string `""`

Constructor cases: If $s \in \{0,1\}^*$ then $s0, s1 \in \{0,1\}^*$. *(arrow to $s0$)* $s \cdot$`"0"`, $s +\!\!+$ `"0"`

In general, Σ^* = finite strings with characters from Σ.

E.g. {`"[","]"`}*: [, [],][[, ...

strings of matched [ ]

[], [][], [[]][]

Define Bkt by

Base case:

Constructor cases:

(Ask for help.)

Base case: $\lambda \in \text{Bkt}$

Constructor cases: If $s, t \in \text{Bkt}$, then $[s], s \cdot t \in \text{Bkt}$.

Or: If $s, t \in \text{Bkt}$, then $[s]t \in \text{Bkt}$.

Is this a complete definition of Bkt?

Problem: $\text{Bkt} = \{\texttt{"["}, \texttt{"]"}\}^*$ satisfies our definition.

Replace "Define Bkt *by" with "Define* Bkt *to be the smallest set such that"*

Replace "Define $\{0,1\}^*$ *by" with "Define* $\{0,1\}^*$ *to be the smallest set s.t."*

(Back on this slide) Fix: define as "smallest set s.t. ..."

Fully parenthesized "+" expressions
$(a+b)$, $(a+(b+a))$, $(((a+b)+(c+d))+a)$, ...
The smallest set S such that
Base case:
Constructor case:

(Beside "Base case") $a, b, c, \ldots \in S$

(Beside "Constructor case") If $e, f \in S$, then $(e+f) \in S$.

Propositional logic formulas with AND and NOT
$(P$ AND $Q)$, (NOT P AND NOT$(Q$ AND $P))$, ...
The smallest set F such that
Base case:
Contsructor cases:

(Beside "Base case") Variables $A, B, \ldots, Z \in F$

(Beside "Constructor case") If $e, f \in F$, then $(e$ AND $f)$, NOT $e \in F$.

Recursively defined set template
Let S be the smallest set such that:
Base case: [list smallest things that are in S]
Constructor cases: If $x, y, \ldots \in S$, then [list ways you can construct new elements out of $x, y, \ldots$]

4 Structural Induction

What if we want to prove a predicate is true for every element of a recursively-defined set? Let's do an example.

Definition of AND-OR formulas, already filled in.
For $e \in F$, let
- $N_v(e) = \#$ occurrences of variables in e
- $N_{op}(e) = \#$ occurrences of operators in e

Prove $\forall e \in F.\ (N_v(e) \leq 1 + N_{op}(e))$.

(Ask how we might do this.)

Proof: by structural induction.

For $e \in F$, let $P(e) =$ "$N_v(e) \leq 1 + N_{op}(e)$)".

Base case: If e is a variable, then $N_v(e) = 1$ and $N_{op}(e) = 0$, so $P(e)$.

Induction step:

Let $e, f \in F$. Assume $P(e)$ and $P(f)$. (I.H.)

AND constructor:

$$\begin{aligned} N_v((e \text{ AND } f)) =& N_v(e) + N_v(f) \\ \leq& (1 + N_{op}(e)) + (1 + N_{op}(f)) \\ =& 2 + N_{op}(e) + N_{op}(f) \end{aligned}$$

$N_{op}((e \text{ AND } f)) = 1 + N_{op}(e) + N_{op}(f)$

So $N_v((e \text{ AND } f)) \leq 1 + N_{op}((e \text{ AND } f))$, i.e. $P((e \text{ AND } f))$.

NOT constructor: $N_v(\text{NOT } e) = N_v(e) \leq 1 + N_{op}(e)$. $1 + N_{op}(\text{NOT } e) = 1 + 1 + N_{op}(e)$. So $N_v(\text{NOT } e) \leq 1 + N_{op}(\text{NOT } e)$, i.e. $P(\text{NOT } e)$.

Structural induction
Given a definition like this: *(Repeat recursively defined set template.)*
To prove $\forall x \in S.\ P(x)$:
- Base case: prove $P(x)$ for each base case $x \in S$.
- Induction steps: For each constructor case, prove P is true for the constructed value assuming it's true for the inputs.

Recursive definition of $\mathbb{N}$
$\mathbb{N}$ is the smallest set such that:
Base case: $0 \in \mathbb{N}$
Constructor case: If $n \in \mathbb{N}$, then $n + 1 \in \mathbb{N}$.
Given $P : \mathbb{N} \to \{\text{T}, \text{F}\}$, what does a structural induction proof of $\forall n \in \mathbb{N}.\ P(n)$ look like?

It's a trick question. It's exactly the same as simple induction.

Base case: $P(0)$

Induction step: Let $n \in \mathbb{N}$. Assume $P(n)$. $\ldots P(n+1)$

Simple induction is a special case of structural induction.