Induction and Recursion
Notes for Phil. 303
JT Fall 2007

Contents

1 Recursion: Some basic facts

A simple pattern of reasoning that shows up from time to time in everyday life, and is indispensible to computer programming and analysis of numbers is *mathematical induction*. There is a related pattern of definition - also very important - called *definition by induction* (or: *definition by recursion*). Typically the uses of this reasoning are relatively abstract, but the core pattern is simple and occurs in concrete cases as well as abstract ones. As we often find in this course, the pattern we are trying to learn is just a common - sense pattern of thinking, rendered in a more rigorous and explicit way.

The basic idea of induction is this. Say you have a fixed starting point and an operation you can apply over and over again. Let's say that when you apply the operation to a given argument, you get something new. Then you can apply this operation to that "something new" and get a further new thing. And you can keep going, applying the operation again and again, and getting new things again and again. (Of course, with some operations you just keep getting the same thing back over and over again.) This gives a way to define a collection: start at the starting point and take what you get by applying the operation over and over again as often as possible. So for example, the set of non-negative integers $\{0, 1, 2, ...\}$ can be characterized by the instruction: start with $\{0\}$, and put into the set everything you get by adding one!

Typically when induction is used, it is applied to infinite sets of abstract things. But the basic reasoning can apply also to finite sets of concrete things. It will be good to consider one mundane example to dispel any air of mystery: the definition of "descendent".

1.1 Inductive definition of "descendent"

A familiar example of an inductive definition (though in everyday conversation it isn't crafted in the canonical form) is the definition of "descendent of x". Recently there was a dispute about whether or not a particular currently living person was a descendent of Thomas Jefferson. But what does it mean to say someone is a descendent of Thomas Jefferson? How do we define that set? (It'll be easier for our exposition if we tweak the definition of "descendent" a bit so that Thomas Jefferson is one of his own descendents.) Informally, we can say: "Thomas Jefferson is one of Jefferson's descendents, and all the children of Jefferson are among Jefferson's descendents, and all the children of each of the children of Jefferson are among Jefferson's descendents, and so on..." Or we might put it a little differently: "You know. You take Jefferson, and his

children, and the children's children, and the children's children's children, and keep on going like that." To the logician, the crucial unclarity is in the meaning of "and so on", or "keep on going like that". Imagine we had to give explicit instructions to a computer to list the Jefferson descendents. How would we make the instructions rigorous?

One way to get oriented is to work backwards:

Under what conditions is Amy one of Jefferson's descendents?
Well, if at least one of her parents is a descendent of Jefferson.

Under what conditions is one of Amy's parents a descendent of Jefferson?
Well, if at least one of their parents is a descendent of Jefferson.

Under what conditions is one of Amy's parents parents a descendent of Jefferson?
⋮

This could in principle generate an infinite regress, except that there is one point where it stops: In some cases the people in question are actually children of *Jefferson*. This "tracing back through the descendents" involves one "fixing the basis" principle "x is a descendent of Jefferson if x is Jefferson" and a further condition that gets repeated over and over again: "x is a descendent of Jefferson if x is the child of a descendent of Jefferson".

We can sum up the definition this way: "x is a descendent of Jefferson if and only if either i) x is Jefferson or ii) x is the child of a descendent of Jefferson." This may appear circular, but it isn't. I'll explain why it isn't circular in a moment. First I'll introduce you to a standard format that we use to express definitions like this:

Basis clause: x is Jefferson.
Recursion clause: x is the child of a descendent of Jefferson.
Closure clause: nothing else is a descendent of Jefferson.

I'll say more about the closure clause later - I'm including it now "just for the record". In fact it is usually just left implicit.

It might seem as if the recursive definition has a crucial, intrinsic problem. One way for a definition to be defective is for it to be *circular*. Say that I want to define "poefool" like this: "x is a poefool if and only if x is a bird and x has feathers, and x is a poefool." This is defective because the term I'm trying to define occurs in the

clause that is supposed to define it. If I find a peacock and ask myself if it is a poefool, I can't use this definition to answer, because I would have to *already* know that it is a poefool to apply the definition. At first glance, the definition of descendent might seem to have this defect, because the term being defined occurs in the definition itself. In fact, this isn't a problem, but you need to understand the structure of recursive definitions to see why not: note that each time you apply the definition, you get a bit closer to the baseline stipulation that Jefferson is a descendent of Jefferson, and that baseline stipulation doesn't presuppose any prior grasp of the definition of "descendent of Jefferson".

Recursive definitions support the special kind of reasoning we've mentioned, called "Inductive Reasoning". The "Jefferson's descendents" example gives a deceptively simple example of this. One test we would regard as conclusive for demonstrating that someone is a Jefferson descendent is a DNA test. We check a sample of Jefferson's DNA and a sample of the prospective descendent's DNA, and see if it matches in the relevant ways. Now why should this tell us anything? Answer: because certain properties of DNA are passed on from parents to children. The reasoning involved is absolutely familiar, but logically it is a bit intricate, so I'll spell it out more ruthlessly than we usually do. I will label the parts of this bit of reasoning so that we can refer to them easily:

Basis: Jefferson has DNA property α.
Induction Step: If a person has DNA property α then their children have DNA property α.
Therefore:
Every descendent of Jefferson has DNA property α.[1]

In most of the inductive arguments we'll study, we actually prove the two premises using mathematics or logic. In this case the two premises are established by empirical scientific investigation of microbiology. But the structure of the reasoning from the premises isn't affected by the way that the premises are justified. There are two observations worth making. First: The inference from the basis and induction step to the conclusion is one we find obvious and untroubling. Second: The inference is logically rather intricate. The premises have to do with the basis and the links of a chain, the conclusion concerns every member of the chain.

[1] Of course, it matters both that Jefferson's descendents have a certain property and that people who aren't Jefferson descendents *don't* have the property. This inductive argument is only relevant to the first of these. But that doesn't matter to the example.

Among the most prominent topics that invite inductive arguments and support recursive definitions are the natural numbers $\{0, 1, 2, \ldots n, \ldots\}$ and languages that are generated by iterating rules. We will be principally concerned with the structured languages, but inductive arguments in arithmetic are simpler, so I'll look at an couple of illustrations from arithmetic as well as from structured languages.

1.2 Recursion on numbers: A simple example

A simple example of a recursive definition is the definition of multiplication by some number (say: 17) in terms of adding 17:

Basis clause: $17 \cdot 0 = 0$
Recursion clause: $17 \cdot (n + 1) = (17 \cdot n) + 17$

(More generally, we can define "$x \cdot n$" recursively by replacing "17" with "x".) Say that we are asked to calculate $17 \cdot 23$. This definition fixes a value for $17 \cdot 23$, though to get it we need to unpack the definition by applying the recursion clause repeatedly until we hit the bedrock of the basis:

$$17 \cdot 23 = (17 \cdot 22) + 17$$

$$= (17 \cdot 21) + 17 + 17$$

$$= (17 \cdot 20) + 17 + 17 + 17$$

$$\vdots$$

$$= (17 \cdot 1) + \underbrace{17 + 17 + 17 \ldots + 17}_{22\,times}$$

$$= (17 \cdot 0) + \underbrace{17 + 17 + 17 \ldots + 17}_{23\,times}$$

$$= (0 + \underbrace{17 + 17 + 17 \ldots + 17}_{23\,times}$$

At each step, until the last one, you get the value of a product of 17 with some number in terms of the product of 17 with a smaller number. At the bottom of the chain, you finally get an expression that is i) unpacked in terms of the basis clause alone, and ii) several "+ 17"'s and doesn't contain any reference to the "$\cdot \cdots 17$" function we are trying to define.

1.3 Induction on numbers: Deal two cards

As noted, when you have a domain of objects structured by a recursive definition, it supports arguments by induction. Since the simplest example of a recursive definition is the definition of the natural numbers in terms of 0 and +1, it is not suprizing that inductive arguments occur especially often in arithmetic.

The standard example of an inductive argument is given in the textbook, section 2.8.1: prove that $1 + 2 + 3 + \ldots n = \frac{n(n+1)}{2}$. Here is another. In some card games, such as blackjack (single deck) or Texas hold 'em poker, at the beginning of a round two cards are dealt to each player. Of course, for the value of the hand it doesn't matter in what order the cards are dealt, if both are face down: the hand with A♠ dealt first and then 6♣ is the same hand as the one with 6♣ first and then A♠. How many possible first hands can there be? The answer is $\frac{(52)\times(51)}{2} = 1326$. This is an instance of a general pattern: whenever *exactly* two things are chosen from a sample of k things, there are exactly $\frac{k(k-1)}{2}$ different possible choices of pairs, if the order doesn't matter. Let's prove the general fact by induction.

Prove: Show that the number of ways to deal two cards out of a deck of k (> 2) cards (ignoring the order in which the cards are dealt, and assuming that all the cards are different) is $\frac{k(k-1)}{2}$.

Proof:
Base Case:
The base case is when $k = 2$. There is, of course, exactly one way of choosing two things from a set of two things. $\frac{2\times(2-1)}{2} = \frac{2}{2} = 1$, so our thesis holds for $k = 2$.

Induction step: Choose $\tilde{k} > 2$ and **assume** [Induction Hypothesis] that there are exactly $\frac{\tilde{k}(\tilde{k}-1)}{2}$ ways of dealing two cards out of a deck of k cards.

We Want to Show (in the Induction Step): There are exactly $\frac{(\tilde{k}+1)\tilde{k}}{2}$ ways of dealing two cards out of a deck of k+1 cards.

Proof (of induction step): Take the deck of $\tilde{k} + 1$ cards and remove one card C^*. The remaining cards, once C^* is removed, are a deck with $\tilde{k}$ cards, and so there are $\frac{\tilde{k}(\tilde{k}-1)}{2}$ ways of dealing two cards from the deck without C^*.

It will be useful to have names for the cards in the deck without C^*; We'll call the cards $C_1, C_2, \ldots, C_{\tilde{k}-1}, C_{\tilde{k}}$.

The deck consisting of $\{C_1, C_2, \ldots, C_{\tilde{k}-1}, C_{\tilde{k}}\}$ has $\tilde{k}$ cards and so by the induction hypothesis, there are exactly $\frac{\tilde{k}(\tilde{k}-1)}{2}$ ways of dealing a pair of cards from the reduced deck. In the full deck $\{C_1, C_2, \ldots, C_{\tilde{k}-1}, C_{\tilde{k}}\} \cup \{C^*\}$ you can deal all the pairs from the smaller deck, *plus* all the pairs consisting of one element from the smaller deck and C^*. Since there are exactly $\tilde{k}$ elements in the smaller deck, there are exactly $\tilde{k}$ different pairs consisting of one element of the smaller deck and C^*. So the full deck contains $\frac{\tilde{k}(\tilde{k}-1)}{2} + \tilde{k}$ possible two-card hands.

Now we just do some simple algebra to get this expression into the form we want:

$\frac{\tilde{k}(\tilde{k}-1)}{2} + \tilde{k} = \frac{\tilde{k}^2-\tilde{k}}{2} + \frac{2\times\tilde{k}}{2} = \frac{\tilde{k}^2-\tilde{k}+(2\times\tilde{k})}{2} = \frac{\tilde{k}^2+\tilde{k}}{2} = \frac{\tilde{k}(\tilde{k}+1)}{2}$

This is just what we want, so the induction step, and hence the thesis, is proven.

1.4 Ordinary Induction and Complete Induction

Proofs by mathematical induction fit into two slightly different schemes, depending on how much you assume about the numbers less than $n + 1$. These argument patterns aren't importantly different, but to avoid confusion it is worthwhile to note the surface difference in form. We have just seen arguments where we prove some property is true of 0 (or whatever is the first in the series - sometimes it is 1), and then we prove that if we assume that property is true of an arbitrary n, it will also be true of $n + 1$. These two supports allow us to conclude that the property holds of every number. A variation is sometimes called the method of *complete induction*, though this usage doesn't appear to be completely standard. In complete induction, we prove the base clause as before, and then we assume that the property holds of *every* number less than $n + 1$. The difference is that instead of assuming the thesis just for the number preceding the given one, we assume it for *every* number less than the given one. Neither of these methods is any less correct than the other: it just happens that one form is convenient for some problems, and the other form is convenient for others. When dealing with strings of symbols in a language, the method of complete induction tends to be a little more useful.

1.5 Recursion and induction on languages with structure

If you speak English (or any other natural language) fluently, you are able to produce long and complicated sentences like:

Incredibly, my neighbor was as irritated as a nest of disturbed bees when I asked reluctantly but insistently for the prompt return of the hedge trimmer I needed to tidy up the back yard before my son's already delayed birthday party.

We can understand sentences like this. Of course, when they are unusually long it may take some concentration to figure out what they say, but we can usually puzzle them out in no more than a few seconds. It's reasonable to conjecture that this ability is grounded in the fact that complex sentences are generated from simpler components by the application (possibly the repeated application) of rules. Here's an example.

Say we have a sentence "The dog bit the cat" of the form "(noun1)(verb)(noun2)". We can form a noun phrase "The cat the dog bit" of the form "The (noun2) the (noun1) (verb)". Also, as our sentence "The dog bit the cat." exemplifies, given two nouns (noun1) and (noun2) and a verb that takes two arguments, we can form a sentence (noun1)(verb)(noun2). Similarly, we can form "The cat chased the rat" from "The cat", "chased" and "the rat", and then from "The cat chased the rat" we can form the noun phrase "The rat the cat chased". Notice that we can iterate these operations over and over: I can form "The dog bit the cat", then form "The cat the dog bit", then taking this as (noun1) I can form "The cat the dog bit chased the rat." And we don't have to stop there. Further iterations give us some real prizes:

The cheese the rat the cat the dog bit chased ate had spoiled in the fridge.

The fridge the cheese the rat the cat the dog bit chased ate had spoiled in sat next to the stove.
⋮

A digression: For entertainment purposes note that some fish, like angler fish, feed themselves by fishing for other fish. We can put this fact this way: Fish fish fish. And of course, following roughly the above pattern, we can note that there are some fish that are the targets of the fish who fish for fish. These are the fish fish fish. So understood, the sentence "Fish fish fish" and the noun phrase "Fish fish fish" support further iterations using the above rules, to get (meaningful) sentences like:

Fish fish fish fish fish fish fish.

The examples we have just spun out are all meaningful English sentences, though it may take some concentration to puzzle out what they mean if this is the first time you've seen them. The way we can understand them is that i) we trace back to the basic meaningful constituents ("cat", "cheese" etc.) and ii) figure out the sentence structure by applying and re-applying the rules. This is recursion in action. As we'll see as the course goes on, this makes it possible learn things about logic and languages by reasoning inductively.

1.6 Closure clause

I should say explicitly what the closure clause does. The closure clause is necessary because there will typically be many sets that contain the basis set and are closed under the conditions given by the recursion clause. We are interested in the *smallest* one. For example, if we make the *huge* assumption for the sake of the example that there was a first pair of human beings, called Adam and Eve, and that every living person is descended from that pair, we can define the set of human beings who have lived or are currently living by this recursive definition:

Basis clause: Adam and Eve are human beings.
Recursion clause: If x is the child of a human being, then x is a human being.

But there are other sets containing Adam and Eve and closed under "x is a child of". For example, the set containing all the human beings who have ever lived, plus the Empire State Building also contains Adam and Eve and is also closed under "x is a child of". The point of the closure clause is to exclude everything - like, in this case, the Empire State Building - that isn't explicitly forced into the set we define by either the basis clause or the recursion clause.

2 Induction on Numbers and Strings: More Cases

2.1 Palindromes

According to the most common definition, a *palindrome* is a string of letters that reads the same way backwards and forwards (ignoring spaces and punctuation). One simple example is "race car". More elaborate examples tend to sound a bit stilted, but they can be found. Some instances are: "Madam, I'm Adam", "A Man, A Plan, A Canal: Panama" and (thought of as spoken by Napoleon before his downfall and exile on the island of Elba): "Able was I, ere I saw Elba." For more, you can search on youtube for a truly odd video by the parodist Weird Al Yankovic making fun of a truly odd Bob Dylan video from the 1960's (Subterranean Homesick Blues). [2] The lyrics of Yankovic's song are a series of palindromes. ("Nurse, I spy gypsies: run!; "Do geese see God?" "Go hang a salami, I'm a lasagna hog." ...)

There are some trivial cases too: "I" is an English language palindrome, and so is "a". For bookkeeping purposes, let's count the empty string as an English language expression: then it is a palindrome as well.

We can study the simpler case of palindromes in a regular language: let's say we are considering the set of all strings in the alphabet $\{a, b, c\}$. As far as the structure of palindromes is concerned, one particular property is especially interesting: if you remove the first and last letter of a palindrome, you get another palindrome, two letters shorter. This fact gives us a way to inductively define "palindrome":

Call a string over $\{a, b, c\}$ an *inductive palindrome* if it satisfies the conditions:

Base clause: ϵ, 'a', 'b', and 'c' are inductive palindromes.
Recursion clause: If σ is an inductive palindrome, then $a * \sigma * a$, $b * \sigma * b$, $c * \sigma * c$ are inductive palindromes.
[Closure: Nothing else is an inductive palindrome in $\{a, b, c\}$].

What we want to do now is **prove** that every palindrome is an inductive palindrome. The proof exploits the recursive structure of "inductive palindrome" to give an inductive proof.

[2] For Yankovic's video, a youtube search on "Yankovic palindrome" will most likely turn it up. For full effect, you might want to watch part of the original Dylan video: A search on "Dylan Subterranean" should do.

Proof:
The induction will be on the length of the string σ.
Base case: Say that σ is 0 symbols long. Then it is ϵ, which is both a palindrome and an inductive palindrome.
Inductive step:
Assume (Induction hypothesis) that every palindrome of length less than k symbols is an inductive palindrome. Say that σ is a palindrome with exactly k symbols.
If σ has exactly one symbol, then it is one of 'a','b', or 'c', which are inductive palindromes by definition. If σ has two or more symbols, then we can remove the first and last letter, leaving a string σ'. Since σ reads the same backwards and forwards, and σ' comes from σ by removing the first and last letter, σ' reads the same backwards as forwards. That is, σ' is a palindrome. Since σ' is a palindrome, and it has fewer than k letters, then it is an inductive palindrome by the induction hypothesis.
Since σ is a palindrome, the first and last letter must be the same, so it is either 'a', 'b', or 'c'. So $\sigma = a * \sigma' * a$, or $\sigma = b * \sigma' * b$, or $\sigma = c * \sigma' * c$, where as we have shown σ' is an inductive palindrome. So, by the inductive clause of the definition of inductive palindrome, σ is an inductive palindrome.

2.2 Fibonacci numbers:

A sequence of numbers that shows up in a surprizing number of places in nature is the Fibonacci sequence:

1,1,2,3,5,8,13,21,34,...

Each number in the sequence is the sum of its two predecessors. The inductive definition of this one is a bit different from the ones we are used to, since we need to consider both $f(k)$ and $f(k-1)$ to define $f(k+1)$, and we need to define two starting points. But these are minor adjustments. Here's the definition:

$f(0) = 1$
$f(1) = 1$
$f(n+1) = f(n-1) + f(n)$ for $n > 1$

As usual, the inductive structure makes it possible to prove things. Here's a question: is the Fibonacci function f we've just defined dominated by the exponential function

$g(x) = 2^x$? That is, is it always the case that $f(x) < g(x)$ (except for $f(0) \leq g(0)$)? Think about how you might prove this. It's easy to confirm this for the base cases:

$f(0) = 1 = 2^0 = g(0)$ and
$f(1) = 1 < 2 = 2^1 = g(0)$

But once you've taken care of these specific cases, how can you go on? The basic observation is simple: use the elementary fact that if $a \leq c$ and $b < c$ then $a + b < 2c$. This tells us that repeatedly taking fibonacci numbers is not going to pull ahead of repeatedly multiplying by two. Try it for the first seven or so values of each sequence:

1,1,2,3,5,8,13...

1,2,4,8,16,32,64,...

No doubt by now you get the idea: the inductive argument is the way to make the idea rigorous. We have already proven the base case for $f(0)$ and $f(1)$ now we **Assume** that for every n such that $1 < n < k$, $f(n) < 2^n$. We want to show $f(k) < 2^k$. That follows immediately from the fact that (by the induction hypothesis) $f(k-1) < 2^{k-1}$ and $f(k-2) < 2^{k-2} < 2^{k-1}$ and the fact that $f(k) = f(k-2)+f(k-1) < 2^{k-1}+2^{k-1} = 2 \times 2^{k-1} = 2^k$.
This proves the inductive step, and we're done.

2.3 Choosing teams:

Say that you have a situation where k friends are scattered among n teams, with both k and n greater than or equal to 4. All the friends want to be on the same team. The league has a rule that you can only move players from one team to another if you choose two players from two *different* teams and transfer them to a third team (different from the first two). Can the friends hope that eventually they might end up on the same team? For any number of teams and friends, is it always possible to repeatedly apply this rule to shift all the friends onto the same team?

Yes it is, and here's the proof:

Proof: This problem illustrates an issue that sometimes comes up with inductive proofs: even if you are sure that an inductive argument is the way to go, it might

be unclear what you should do the induction *on*. Here we have the choice between induction on the number of players or the number of teams. In this situation you should fiddle around a bit to figure out what works, and do the induction on the number of players.

Base case: Say that n, the number of players, = 4. This is one of the rare cases where the base case is more involved than the induction step. Take the teams with our friends on them, plus as many extra teams we need to get a total of four. Ignore the other teams. We'll write (n_1, n_2, n_3, n_4) ($n_1, n_2, n_3 and n_4$ all numbers) to represent the number of friends on each team. So, for example (2, 1, 0, 1) means that 2 friends are on team 1, one of the friends is on team 2, one of the friends is on team 4, and none of the friends is on team 3. Here are the possible starting configurations; the last of them represents the target situation where all the friends are on the same team:

Configurations:
a) (1,1,1,1)
b) (1,2,1,0)
c) (2,2,0,0)
d) (3,1,0,0)
e) (4,0,0,0)

(Note that the order doesn't matter, so we can count both (say) (1,2,1,0) and (1,1,0,2) as configuration b))

The hardest case is a). Beginning there, we can move to e) in a way that passes through every other configuration. So this sequence of moves also shows how to cover the other configurations:

$$(1,1,1,1) \rightarrow \underbrace{(3,1,0,0)}_{configuration\ d} \rightarrow \underbrace{(2,0,2,0)}_{configuration\ c} \rightarrow \underbrace{(1,0,1,2)}_{configuration\ b} \rightarrow \underbrace{(0,0,0,4)}_{configuration\ e}$$

So however the four friends may start out distributed, you can shift them onto the same team while obeying the "move two from different teams onto a third team" rule.

Induction step:
OK, so what should we do if we have more than four players? The induction step will show that if we have a procedure for k players, then we have a procedure for k+1.

Assume: (Induction Hypothesis) Any distribution of k players can be reconfigured by repeatedly using the "choose from two onto a third" rule so that all k players eventually end up on the same team. Say that we have k+1 players. Choose one of the players - call her Jill. The collection of all the players besides Jill is a collection of k players, and so by the induction hypothesis we can rearrange them onto the same team. Do that. If they are on the same team as Jill we are done. If not, then we have the following configuration (again, ignoring extra teams): (1,k,0,0).

Here is how to rearrange the players:

$(1, k, 0, 0) \to (0, k-1, 2, 0) \to (0, k-2, 1, 2) \to (2, k-3, 0, 2) \to (1, k-1, 0, 1) \to (0, k+1, 0, 0)$

This completes the induction step, so we are done.

2.4 Something to try yourself:

Here is the inductive definition of n! (read "n factorial"):
$1! = 1$
$(n+1)! = (n+1) \times n!$

That is, $n! = \underbrace{(n \times (n-1) \times \ldots 3 \times 2 \times 1)}_{n \text{ times}}$

Exercise:

Prove by induction: For every n greater than 3, $2^n < n! < n^n$.