Inductive data

Robert Y. Lewis

CS 0220 2024

April 24, 2024

Overview

Induction on $\mathbb{N}$

We introduced induction as a technique to prove things about natural numbers.

It makes some intuitive sense. But let's examine things more carefully.

Defining $\mathbb{N}$

What are the natural numbers?

1. 0 is a natural number.
2. For any natural number k, $k+1$ is a natural number. successor(k)
3. successor is injective.
4. For every k, successor(k) $\neq 0$.
5. Every natural number is either 0 or the successor of another natural number.

Are there any sets that satisfy properties 1-5 that *don't* look like $\mathbb{N}$?

Defining $\mathbb{N}$

Let's try again.

1. 0 is a natural number.
2. For any natural number k, $k + 1$ is a natural number. successor(k)
3. successor is injective.
4. For every k, successor(k) $\neq 0$.
5. Every natural number can be represented as a (finite) directed tree, where each node is either
   - labeled 0, and has no children; or
   - labeled successor, and has one child.

Condition 5 is equivalent to the principle of induction.

Again, succinctly

We define $\mathbb{N}$ to be an *inductive set* with *constructors*

- $0 : \mathbb{N}$
- successor: $\mathbb{N} \to \mathbb{N}$.

An inductive set is defined by giving a list of constructors that are assumed to satisfy properties 3-5.
See also: *inductive types* or *algebraic data types* in some programming languages.

Inductive sets are sets of “discrete objects.”

And, recursion

Let A be any set, $a \in A$, and $g : \mathbb{N} \times A \to A$. There exists a unique function $f : \mathbb{N} \to A$ satisfying the two clauses:

- $f(0) = a$
- $f(k+1) = g(k, f(k))$

“Exists” and “unique.” In other words: “to define a function with domain $\mathbb{N}$, we can describe its behavior on the two constructors.”

Sounds a lot like induction. And the tree property.

Inductive lists

Let A be a set. The set $L(A)$ of lists of elements of A is an inductive set with constructors

- $\mathrm{nil} : L(A)$
- $\mathrm{cons} : A \times L(A) \to L(A)$

"To create a list, either create the empty list, or take a list and tack on one more value."

Induction on lists

- nil : $L(A)$
- cons : $A \times L(A) \to L(A)$

Tree property?

Induction principle?

To show that $P(l)$ holds for every list $l \in L(A)$, show:

- $P(nil)$
- For every $a \in A$ and $l \in L(A)$, $P(l) \to P(cons(a, l))$

Inductive integers?

Let's try to define $\mathbb{Z}$ as an inductive set.

Constructors:

- $0 : \mathbb{Z}$
- successor : $\mathbb{Z} \to \mathbb{Z}$
- predecessor : $\mathbb{Z} \to \mathbb{Z}$

Fails: why?

Inductive integers!

A working, if lame, attempt:

Constructors:

- $0 : \mathbb{Z}$
- posOfNat : $\mathbb{N} \to \mathbb{Z}$
- negOfNat : $\mathbb{N} \to \mathbb{Z}$

posOfNat(n) “=” $n + 1$
negOfNat(n) “=” $-(n + 1)$

Inductive formulas

The set F of formulas of propositional logic is an inductive set with constructors

- $\mathit{letter} : \mathbb{N} \to F$
- $\mathit{not} : F \to F$
- $\mathit{and} : F \times F \to F$
- $\mathit{or} : F \times F \to F$
- $\mathit{implies} : F \times F \to F$
- $\mathit{iff} : F \times F \to F$

Principle of induction? To prove $P(\varphi)$ holds for every prop formula φ, it suffices to show:

- $P(\mathit{letter}(i))$ for every i ("P holds of every propositional letter")
- $P(\varphi) \to P(\mathit{not}(\varphi))$
- $P(\varphi_1) \wedge P(\varphi_2) \to P(\mathit{and}(\varphi_1, \varphi_2))$
- $P(\varphi_1) \wedge P(\varphi_2) \to P(\mathit{or}(\varphi_1, \varphi_2))$
- ...

Inductive formulas

Recursion on formulas, in words:

To define a function $f : F \to A$, it suffices to describe the behavior of F on each constructor of F.

Example: evaluation $E(\varphi)$ under a propositional assignment $v : \mathbb{N} \to \{T, F\}$.

- $E(letter(i)) = v(i)$
- $E(not(\varphi)) = NOT(E(\varphi))$
- $E(and(\varphi_1, \varphi_2)) = AND(E(\varphi_1), E(\varphi_2))$

Challenge: phrase this like we phrased recursion on $\mathbb{N}$.

Proofs as data

We have a technique for figuring out if a propositional formula is *valid*: write the truth table, see if all columns are T.

This is more of a “process” than an “object.” Intuition: if you write down an argument like this, the only way I can check it is by doing it myself and comparing.

Other ways?

Proofs as data: introduction rules

How can I prove $\varphi_1 \wedge \varphi_2$? Prove φ_1 and then prove φ_2.

How can I prove $\varphi_1 \vee \varphi_2$? Prove φ_1. Alternatively, prove φ_2.

This sounds sort of inductive. Constructors?

- $\mathit{and\_intro} : \mathit{proof}(\varphi_1) \times \mathit{proof}(\varphi_2) \to \mathit{proof}(\varphi_1 \wedge \varphi_2)$
- $\mathit{or\_intro\_left} : \mathit{proof}(\varphi_1) \to \mathit{proof}(\varphi_1 \vee \varphi_2)$
- $\mathit{or\_intro\_right} : \mathit{proof}(\varphi_2) \to \mathit{proof}(\varphi_1 \vee \varphi_2)$

Proofs as data: elimination rules

What can I do with a proof of $\varphi_1 \wedge \varphi_2$? Prove φ_1. Alternatively, prove φ_2.

- $\mathit{and\_elim\_left} : \mathit{proof}(\varphi_1 \wedge \varphi_2) \to \mathit{proof}(\varphi_1)$
- $\mathit{and\_elim\_right} : \mathit{proof}(\varphi_1 \wedge \varphi_2) \to \mathit{proof}(\varphi_2)$

What can I do with a proof of $\varphi_1 \vee \varphi_2$? Case split…tricky.

Need to analyze implication first, which also muddies the picture a bit.

For another time

Can't dive into the details now. But we can make things more or less work.

Induction on proofs?? "I can only construct proofs of valid formulas."

Recursion on proofs?? Given a proof, reconstruct the formula it proves—proof checking!

Proofs are directed trees!

Final thoughts

We've seen a lot of topics this semester. Remember why we've done this:

- Vocabulary. Use the languages of logic, combinatorics, probability, ... as a shared, precise vocabulary for discussing problems.
- Abstraction. A lot of the problems we've studied will show up in different contexts, in and out of computer science. Remember our abstract solutions and adapt them to reality.
- Team problem solving. CS is collaborative, and hopefully you've gotten practice solving problems with a team.