Math 321: A return to induction

Kameryn J Williams

University of Hawaiʻi at Mānoa

Spring 2021

Games

- We've been talking about games. The big theorem we worked up to is that for a broad class of games—two player, perfect information, non-random, finite—that always one player has a winning strategy (or both can force a draw, if draws are allowed).
- The proof went by looking at game trees.
  - We can think of all possible plays of the game as forming a tree.
  - Each terminal position in the tree is labeled with who wins if the game reaches that position.
  - We can then back-propagate this information to the tree, determining for each position who is in a winning position—if they play right, they can always force a win.
  - The winning strategy for the game is then to play according to these labels to stay in a winning position.

Games

- We've been talking about games. The big theorem we worked up to is that for a broad class of games—two player, perfect information, non-random, finite—that always one player has a winning strategy (or both can force a draw, if draws are allowed).
- The proof went by looking at game trees.
  - We can think of all possible plays of the game as forming a tree.
  - Each terminal position in the tree is labeled with who wins if the game reaches that position.
  - We can then back-propagate this information to the tree, determining for each position who is in a winning position—if they play right, they can always force a win.
  - The winning strategy for the game is then to play according to these labels to stay in a winning position.
- This construction is done inductively on the game tree.

Games

- We've been talking about games. The big theorem we worked up to is that for a broad class of games—two player, perfect information, non-random, finite—that always one player has a winning strategy (or both can force a draw, if draws are allowed).
- The proof went by looking at game trees.
  - We can think of all possible plays of the game as forming a tree.
  - Each terminal position in the tree is labeled with who wins if the game reaches that position.
  - We can then back-propagate this information to the tree, determining for each position who is in a winning position—if they play right, they can always force a win.
  - The winning strategy for the game is then to play according to these labels to stay in a winning position.
- This construction is done inductively on the game tree.

So let's talk about why it's valid to do induction on more general structures than just the natural numbers.

Induction

$\mathbb{N}$ is the main setting where mathematicians use induction, but it's not the only one.

- Some examples come from computer science.
- For example, computer scientists and programmers often are interested in trees. Many kinds of data naturally can be represented as trees.
- If you want to prove something is true for all nodes in a tree, induction is one method available to you.

Induction

$\mathbb{N}$ is the main setting where mathematicians use induction, but it's not the only one.

- Some examples come from computer science.
- For example, computer scientists and programmers often are interested in trees. Many kinds of data naturally can be represented as trees.
- If you want to prove something is true for all nodes in a tree, induction is one method available to you.
- Another computer science example is formal grammars. You want to describe some formal language, such as a programming language, by giving recursive rules for when something is a valid piece of syntax.
- To then prove things about this formal language, you can do induction, with the recursive rules corresponding to the inductive steps in the argument.

Induction

$\mathbb{N}$ is the main setting where mathematicians use induction, but it's not the only one.

- Some examples come from computer science.
- For example, computer scientists and programmers often are interested in trees. Many kinds of data naturally can be represented as trees.
- If you want to prove something is true for all nodes in a tree, induction is one method available to you.
- Another computer science example is formal grammars. You want to describe some formal language, such as a programming language, by giving recursive rules for when something is a valid piece of syntax.
- To then prove things about this formal language, you can do induction, with the recursive rules corresponding to the inductive steps in the argument.

It's good to understand how induction works on $\mathbb{N}$ as the most important case. But you should think of induction as a more general principle. If you're looking at any object built up according to recursive rules, you can use induction to prove facts about that object.

The least number principle

With $\mathbb{N}$, we saw that the least number principle was an equivalent principle to induction.

- If there is a natural number satisfying some property, there is a *smallest* natural number satisfying that property.
- If $X \subseteq \mathbb{N}$ is nonempty, then X has a least element.

The least number principle

With $\mathbb{N}$, we saw that the least number principle was an equivalent principle to induction.

- If there is a natural number satisfying some property, there is a *smallest* natural number satisfying that property.
- If $X \subseteq \mathbb{N}$ is nonempty, then X has a least element.

To use the LNP to show induction is valid, the idea is to look at minimal counterexamples.

The least number principle

With $\mathbb{N}$, we saw that the least number principle was an equivalent principle to induction.

- If there is a natural number satisfying some property, there is a *smallest* natural number satisfying that property.
- If $X \subseteq \mathbb{N}$ is nonempty, then X has a least element.

To use the LNP to show induction is valid, the idea is to look at minimal counterexamples.

- Consider a set X of natural numbers.
- Suppose we know that $0 \in X$ and that $n \in X$ implies $n + 1 \in X$.
- Why can we conclude that $X = \mathbb{N}$?

The least number principle

With $\mathbb{N}$, we saw that the least number principle was an equivalent principle to induction.

- If there is a natural number satisfying some property, there is a *smallest* natural number satisfying that property.
- If $X \subseteq \mathbb{N}$ is nonempty, then X has a least element.

To use the LNP to show induction is valid, the idea is to look at minimal counterexamples.

- Consider a set X of natural numbers.
- Suppose we know that $0 \in X$ and that $n \in X$ implies $n + 1 \in X$.
- Why can we conclude that $X = \mathbb{N}$?
- Suppose toward a contradiction that $X \neq \mathbb{N}$.
- By the LNP, $\mathbb{N} \setminus X$ has a least element m.
- It cannot be that $m = 0$, because $0 \in X$. So $m = n + 1$ for some $n \in X$. But then $m \in X$. Contradiction.

Generalizing the least number principle

- Some ordered structures, like $\mathbb{N}$, are totally ordered (synonymously: linearly ordered): if you take any two natural numbers, either they're the same or one is ordered above the other.
- Other ordered structures, like trees, are merely partially ordered: you can have two different elements in the order which are incomparable, neither above nor below the other.

Generalizing the least number principle

- Some ordered structures, like $\mathbb{N}$, are totally ordered (synonymously: linearly ordered): if you take any two natural numbers, either they're the same or one is ordered above the other.
- Other ordered structures, like trees, are merely partially ordered: you can have two different elements in the order which are incomparable, neither above nor below the other.
- So for this more general setting, we don't want to talk about least or smallest elements, we need a new concept.

Generalizing the least number principle

- Some ordered structures, like $\mathbb{N}$, are totally ordered (synonymously: linearly ordered): if you take any two natural numbers, either they're the same or one is ordered above the other.
- Other ordered structures, like trees, are merely partially ordered: you can have two different elements in the order which are incomparable, neither above nor below the other.
- So for this more general setting, we don't want to talk about least or smallest elements, we need a new concept.
- In an ordered structure, an element m is minimal if there is nothing smaller than it. (But there may be elements off to the side which are incomparable.)

The minimal object principle

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Synonymously, we say that X is well-founded.

The minimal object principle

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Synonymously, we say that X is well-founded.

Trees where every path is finite, like the game trees we considered, satisfy the minimal object principle.

Let's see why that is.

Consider a tree where every path is finite.

The minimal object principle

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Synonymously, we say that X is well-founded.

Trees where every path is finite, like the game trees we considered, satisfy the minimal object principle.

Let's see why that is.

Consider a tree where every path is finite.

- Suppose toward a contradiction that this tree doesn't satisfy the MOP. Then, there is some nonempty subset Y of the tree which has no minimal elements.

The minimal object principle

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Synonymously, we say that X is well-founded.

Trees where every path is finite, like the game trees we considered, satisfy the minimal object principle.

Let's see why that is.

Consider a tree where every path is finite.

- Suppose toward a contradiction that this tree doesn't satisfy the MOP. Then, there is some nonempty subset Y of the tree which has no minimal elements.
- Pick any element x_0 of Y. Because Y has no minimal elements, there is a an element in Y which is smaller than x_0.
- So pick x_1 to be an element of Y which is smaller than x_0.

The minimal object principle

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Synonymously, we say that X is well-founded.

Trees where every path is finite, like the game trees we considered, satisfy the minimal object principle.

Let's see why that is.

Consider a tree where every path is finite.

- Suppose toward a contradiction that this tree doesn't satisfy the MOP. Then, there is some nonempty subset Y of the tree which has no minimal elements.
- Pick any element x_0 of Y. Because Y has no minimal elements, there is a an element in Y which is smaller than x_0.
- So pick x_1 to be an element of Y which is smaller than x_0.
- And we can continue this process. Having already picked $x_0 > x_1 > \cdots > x_n$ from Y, pick x_{n+1} from Y to be smaller than x_n.

The minimal object principle

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Synonymously, we say that X is well-founded.

Trees where every path is finite, like the game trees we considered, satisfy the minimal object principle.

Let's see why that is.

Consider a tree where every path is finite.

- Suppose toward a contradiction that this tree doesn't satisfy the MOP. Then, there is some nonempty subset Y of the tree which has no minimal elements.
- Pick any element x_0 of Y. Because Y has no minimal elements, there is a an element in Y which is smaller than x_0.
- So pick x_1 to be an element of Y which is smaller than x_0.
- And we can continue this process. Having already picked $x_0 > x_1 > \cdots > x_n$ from Y, pick x_{n+1} from Y to be smaller than x_n.
- So we have constructed an infinite path through the tree, a contradiction.

From the minimal object principle to induction

Just like the least number principle for $\mathbb{N}$ implies induction is valid, the minimal object principle for an ordered structure X implies induction is valid for X.

But like with going from the least number principle to the minimal object principle, we have to formulate things a bit differently for the more general setting.

The new complication is: In $\mathbb{N}$ each number is either 0, the smallest number, or else it has exactly one immediate predecessor. In a more general setting, an element might have many immediate predecessors. So it's not enough to just look at one previous case, we need to look at all previous cases.

From the minimal object principle to induction

Just like the least number principle for $\mathbb{N}$ implies induction is valid, the minimal object principle for an ordered structure X implies induction is valid for X.

But like with going from the least number principle to the minimal object principle, we have to formulate things a bit differently for the more general setting.

The new complication is: In $\mathbb{N}$ each number is either 0, the smallest number, or else it has exactly one immediate predecessor. In a more general setting, an element might have many immediate predecessors. So it's not enough to just look at one previous case, we need to look at all previous cases.

So to generalize this, we want to use the strong induction formulation.

From the minimal object principle to induction

Just like the least number principle for $\mathbb{N}$ implies induction is valid, the minimal object principle for an ordered structure X implies induction is valid for X.

But like with going from the least number principle to the minimal object principle, we have to formulate things a bit differently for the more general setting.

The new complication is: In $\mathbb{N}$ each number is either 0, the smallest number, or else it has exactly one immediate predecessor. In a more general setting, an element might have many immediate predecessors. So it's not enough to just look at one previous case, we need to look at all previous cases.

So to generalize this, we want to use the strong induction formulation.

Induction for a general ordered structure X:

- Phrased in terms of sets: Consider a subset Y of X. Suppose that Y satisfies the property that for any $x \in X$, if every $y < x$ is in Y, then $x \in Y$. Then, $Y = X$.
- Or you to phrase it in terms of a proof strategy. To prove that every $x \in X$ satisfies some property $P(x)$, it is enough to do the following:
  - Consider an arbitrary $x \in X$.
  - Assume that every $y < x$ satisfies $P(y)$.
  - Prove $P(x)$.

From the minimal object principle to induction

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Induction is valid for X if it satisfies the following for every $Y \subseteq X$: Suppose that Y satisfies the property that for any $x \in X$, if every $y < x$ is in Y, then $x \in Y$. Then, $Y = X$.

From the minimal object principle to induction

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Induction is valid for X if it satisfies the following for every $Y \subseteq X$: Suppose that Y satisfies the property that for any $x \in X$, if every $y < x$ is in Y, then $x \in Y$. Then, $Y = X$.

Let's see that the MOP for X implies induction is valid for X. We follow the same argument as showing that the LNP for $\mathbb{N}$ implies induction is valid for $\mathbb{N}$.

From the minimal object principle to induction

Consider an ordered structure X.

- X satisfies the minimal object principle if given any nonempty subset $Y \subseteq X$, there is a minimal element of Y (possibly more than one).
- Induction is valid for X if it satisfies the following for every $Y \subseteq X$: Suppose that Y satisfies the property that for any $x \in X$, if every $y < x$ is in Y, then $x \in Y$. Then, $Y = X$.

Let's see that the MOP for X implies induction is valid for X. We follow the same argument as showing that the LNP for $\mathbb{N}$ implies induction is valid for $\mathbb{N}$.

- Consider $Y \subseteq X$, and suppose that for any $x \in X$, if every $y < x$ is in Y, then $x \in Y$. We want to see that $Y = X$.
- Assume toward a contradiction that $Y \neq X$. That is, $X \setminus Y$ is nonempty.
- By the MOP, $X \setminus Y$ has a minimal element, call it m.
- By definition of minimality, if $y < m$ in X, then $y \in Y$. This holds for all $y < m$.
- So by the assumption on Y we get that $m \in Y$. Contradiction.

Summing things up

Because game trees satisfy the MOP, a special case of what we just saw is that induction is valid for game trees (and, more generally, any tree whose paths are all finite). So the inductive argument to label the winning positions in a game tree is valid.

Summing things up

Because game trees satisfy the MOP, a special case of what we just saw is that induction is valid for game trees (and, more generally, any tree whose paths are all finite). So the inductive argument to label the winning positions in a game tree is valid.

In fact, the opposite direction also holds: if induction is valid for an ordered structure X, then X satisfies the minimal object principle.

Summing things up

Because game trees satisfy the MOP, a special case of what we just saw is that induction is valid for game trees (and, more generally, any tree whose paths are all finite). So the inductive argument to label the winning positions in a game tree is valid.

In fact, the opposite direction also holds: if induction is valid for an ordered structure X, then X satisfies the minimal object principle.

But this is more difficult, and we don't need it, so I won't give you a proof.

Summing things up

Because game trees satisfy the MOP, a special case of what we just saw is that induction is valid for game trees (and, more generally, any tree whose paths are all finite). So the inductive argument to label the winning positions in a game tree is valid.

In fact, the opposite direction also holds: if induction is valid for an ordered structure X, then X satisfies the minimal object principle.

But this is more difficult, and we don't need it, so I won't give you a proof.

To sum up, here are three ways you can think about induction:

- It's a property about the natural numbers, giving you a proof strategy to prove universal statements about $\mathbb{N}$.
- It's a property about any structure built up according to recursive rules.
- It's a property about any structure satisfying the minimal object property.